Overview

Abstract

This document specifies the ICQ (I nterC hainQ ueries) module for the Neutron network.

The ICQ module implements a mechanism to retrieve data from remote chains connected to Neutron via IBC.

Concepts

A smart-contract can register two types of Interchain Query for particular chain with some query payload andupdate_period :

- Key-Value query (KV-query) - to readvalues
- from Cosmos-SDK KV-storage on remote chain which are stored under a set ofkeys
- ;
- Transactions query (TX-query) - find transactions on remote chain under by condition (transactions filter).

⚠IMPORTANT NOTE ABOUT KV-QUERIES

Due to abug in ICS23 package, it's currently impossible to query an empty ornil value from a remote chain.

Meaning if your KV-query is registered with keyK and a value under this key isnil or empty, submission of such KV result will fail due to IAVL-proof verification error:failed to verify proof: empty value in membership proof

Moreover, due tothe nature of IAVL-proofs (which is an underlying mechanism of verification of a validity of KV-queries results), it's also impossible to verify IAVL-absence proof if it contains IAVL-proof withnil or empty value:failed to verify proof: could not verify absence of key. Please ensure that the path is correct.

We are in contact with ics23 team to fix this issue ASAP, but in the meantime the only way to deal with the bug - is to avoid querying keys withnil or empty values. ICQ Relayer keeps track of registered Interchain Queries by querying all existed ICQs at the start of work and by subscribing onUpdate andDelete events which are emitted in corresponding Neutron handlers. When the ICQ Relayer sees that it's time to perform an interchain query, it makes a necessary RPC call to a remote chain and makes the results available for the Neutron's smart contracts by submitting the result to the module. Read more about it at theRelayer's page .

Neutron verifies the data and processes the query result depending on the interchain query type:

- in case of a KV-query, the ICQ module saves the result into module's storage, and passed the query id to the contract'sSudoKVQueryResult
- handler
- ;
- in case of a TX-query, the ICQ moduledoes not
- save the result to the storage, finds the contract that registered the query,
- and passes the full result to the contract'sSudoTXQueryResult
- handler
- .

Query creation deposit

In order to clean up ledger from not used, outdated queries special deposit mechanism is usedRegisteredQuery containsdeposit field, this field is used to collect escrow payment for query creation. In order to return escrow payment aRemoveInterchainQuery message should be issued.

Permission to performRemoveInterchainQuery message is based on three parameters:

1. query_submit_timeout
2. — a module parameter which can be thought of as query service period;
3. last_submitted_result_local_height
4. — registered query's property representing the Neutron's height the query was updated last time at;
5. registered_at_height
6. — registered query's property representing the Neutron's height the query was registered at.

The permissions to executeRemoveInterchainQuery are as follows:

- within the service period (i.e. ifcurrent_height <= last_submitted_result_local_height + query_submit_timeout && current_height <= registered_at_height + query_submit_timeout
-) only the query's owner is permissioned to remove it;
- beyond the service period (i.e. ifcurrent_height > last_submitted_result_local_height + query_submit_timeout ||

current_height > registered_at_height + query_submit_timeout
-) anyone can remove the query and take the deposit as a reward.

Amount of coins to deposit is defined via parameter (query_deposit) controlled by governance proposal.

In other words, it is expected of the query owner to remove its queries when they are not needed anymore. If a query hasn't been in use for thequery_submit_timeout and owner hasn't removed it, network users are granted with an opportunity to clean the chain up and raise assets for it.

Transaction filters

Since events themselves are not part of the consensus and are not included in the transaction result, it's necessary toimplement additional checks in yourSudoTXQueryResult handler to check that result transactions satisfies your transactions filter. For instance, you can check that messages in the transactions have proper types, payload, etc. If your contract does not have such checks, malicious relayer can send a fully valid Tendermint transaction which does not satisfy your defined transactions filter, and your business-logic can be broken.

NOTE: when registering a TX-query, you write the transaction filters as filters for transaction events. When you check the submitted transaction in your contracts, though, you can only check the information that is stored on-chain (i.e., message fields for messages in a transaction). To put it another way, the set of values that you can use to filter transactions is the intersection of the values that are added to transaction events (used by the ICQ relayer to perform the search) and the values included directly to sdk.Msgs (can be used by your code to check whether the submitted transaction matches your query). You can see more info, examples and recommendations about proper transactions result handlinghere .

[{ "field" :

"{eventType}.{attributeKey}" ,

"val" :

"{attributeValue}" ,

"op" :

"gte" } , ...] Maximum allowed amount of filters is 32. Supplying more filters than allowed will return an error.

Supported operators:

- eq
- lt
- gt
- lte
- gte

The ICQ relayer can easily parse this format and compose it into usualTendermint syntax for searching transactions.

Suppose you want to search for transactions that meet specific conditions on a remote chain. You can build a query using various filters to narrow down the search.

Finding Transfer Transactions with a Specific Amount

[{ "field" :

"transfer.amount" ,

"op" :

"eq" ,

"val" :

1000 }] This filter searches for all transfer transactions with an exact amount of 1000. The ICQ relayer converts it into the Tendermint search string:

"transfer.amount" = 1000

Searching for Transactions within a Range of Dates

[{ "field" :

"timestamp" ,

"op" :

"gte" ,

"val" :

"2023-01-01T00:00:00Z" } ,

{ "field" :

"timestamp" ,

"op" :

"lt" ,

"val" :

"2023-02-01T00:00:00Z" }] This filter queries for all transactions that were processed between January 1, 2023, and February 1, 2023. The corresponding Tendermint search string would be:

"timestamp" >= "2023-01-01T00:00:00Z" AND "timestamp" < "2023-02-01T00:00:00Z"

Combining Multiple Conditions

[{ "field" :

"message.module" ,

"op" :

"eq" ,

"val" :

"bank" } ,

{ "field" :

"transfer.sender" ,

"op" :

"eq" ,

"val" :

"neutron1suhgf5svhu4usrurvxzlgn54ksxmn8gljarjtxqnapv8kjnp4nrstdxvff" } ,

{ "field" :

"transfer.amount" ,

"op" :

"gt" ,

"val" :

500 }] This example searches for bank transfer transactions sent by a specific address (neutron1suhgf5svhu4usrurvxzlgn54ksxmn8gljarjtxqnapv8kjnp4nrstdxvff) and with an amount greater than 500. The search string would be:

"message.module" = "bank" AND "transfer.sender" = "neutron1suhgf5svhu4usrurvxzlgn54ksxmn8gljarjtxqnapv8kjnp4nrstdxvff" AND "transfer.amount" > 500

Effects of Filters

The filters in thetransactions_filter field allow for refined and targeted querying of transactions. Some effects of applying these filters are:

- Increased Efficiency
- : By narrowing down the search criteria, the query can return results more quickly, reducing the computational

resources required.
- Improved Relevance
- : Filters ensure that only transactions that meet specific criteria are returned, making the results more relevant to the user's needs.
- Flexibility
- : Users can combine different operators and fields to create complex queries that match their exact requirements.
- Error Handling
- : Providing incorrect or conflicting filters might result in an error, so the filter structure must be carefully constructed to avoid issues.

By understanding the usage of thetransactions_filter field, developers and users can leverage the power of targeted querying to interact with remote chains in a more effective and efficient manner.

Having Fewer or More Filters

Fewer Filters

Pros:

- Broader Results
- : Using fewer filters will generally lead to a larger result set, capturing more transactions that meet broad criteria.
- Faster Execution
- : With less complexity, the query may execute more quickly, as there are fewer conditions to evaluate.

Cons:

- Less Precision
- : Fewer filters may lead to less relevant results if the query is too broad.

More Filters

Pros:

- More Specific Results
- : More filters allow for more targeted and precise queries, narrowing down the result set to only the most relevant transactions.
- Enhanced Control
- : More filters offer greater control over the query, enabling more complex and nuanced searches.

Cons:

- Slower Execution
- : More complex queries with multiple filters may take longer to execute, as each additional condition adds to the computational load.
- Potential Overfitting
- : Too many filters may lead to an overly narrow search, missing relevant transactions or even resulting in no results at all if the filters are too restrictive.

Good Practices

1. Start with Core Criteria
2. : Identify the essential criteria for your query and start with those filters. It helps to focus on what you really need from the results.
3. Incrementally Refine
4. : If needed, add additional filters incrementally to refine the results, testing at each stage to ensure relevance.
5. Avoid Redundancy
6. : Ensure that each filter adds value to the query and that there are no redundant or conflicting filters.
7. Test Performance
8. : Consider testing the query with different numbers of filters to gauge performance and result relevance, especially if using many filters.
9. Use the Maximum Limit Wisely
10. : Note that the maximum allowed amount of 32 filters is a technical constraint.

How Many Filters Do You Need?

The optimal number of filters depends on the specific use case and the balance between precision and performance. Generally, it's best to use the minimum number of filters that provide the necessary specificity for your query. Using too few may yield irrelevant results, while using too many may overly narrow the search or impact performance.

Conclusion

The number of filters in a query is a vital consideration, influencing both the relevance of the results and the performance of the query. Striking the right balance requires a thoughtful approach, considering the specific needs of the query, and adhering to good practices for constructing and refining filters. Previous Events Next Messages