Spec Reference Guide

This guide provides a detailed reference to the various specifications within the Lava Network. It encompasses the structure and definitions of proposals, specs, API collections, service APIs, and associated extensions. The objective is to ensure that developers, validators, and other stakeholders have a clear and consistent understanding of the configurations and functionalities.

📌 File Structure

▼ 🔝 Tree Structure

```
Spec (JSON)
│
├── Proposal (`proposal`)
│   ├── title
│   ├── description
│   │
│   └── Specifications (`specs`)
│       ├── index
│       ├── name
│       ├── enabled
│       ├── imports
│       ├── reliability_threshold
│       ├── data_reliability_enabled
│       ├── block_distance_for_finalized_data
│       ├── blocks_in_finalization_proof
│       ├── average_block_time
│       ├── allowed_block_lag_for_qos_sync
│       ├── min_stake_provider
│       └── min_stake_client
│       │
│       └── API Collections (`api_collections`)
│           ├── enabled
│           ├── collection_data
│           │   ├── api_interface
│           │   ├── internal_path
│           │   ├── type
│           │   └── add_on
│           ├── Service APIs (`apis`)
│           │   ├── name
│           │   ├── block_parsing
│           │   │   ├── parser_arg
│           │   │   └── parser_func
│           │   ├── compute_units
│           │   ├── enabled
│           │   ├── category
│           │   │   ├── deterministic
│           │   │   ├── local
│           │   │   ├── subscription
│           │   │   └── stateful
│           │   └── extra_compute_units
│           ├── headers
│           ├── inheritance_apis
│           ├── parse_directives
│           ├── Verifications (`verifications`)
│           │   ├── name
│           │   └── values
│           └── Extensions (`extensions`)
│               ├── name
│               ├── cu_multiplier
│               └── rule
│
└── Deposit (`deposit`)
    └── deposit
```

JSON (Template)

```
{
  "proposal": {
    "title": "Add Specs: API/Chain ",
    "description": "...",
    "specs": {
      "index": "NAME",
      "name": "name of the chain/api",
      "enabled": true,
      "imports": [],
      "reliability_threshold": 268435455,
      "data_reliability_enabled": true,
      "block_distance_for_finalized_data": 0,
      "blocks_in_finalization_proof": 1,
      "average_block_time": 0,
      "allowed_block_lag_for_qos_sync": 1,
      "min_stake_provider": {
        "denom": "ulava",
        "amount": "5000000000"
      },
      "min_stake_client":  {
        "denom": "ulava",
        "amount": "5000000000"
      },
      "api_collections": [
        {
          "enabled": true,
          "collection_data": {
            "api_interface": "",
            "internal_path": "",
            "type": "",
            "add_on": ""
          },
          "apis": [
            {
              "name": "",
              "block_parsing": {
                "parser_arg": [
                  "latest"
                ],
                "parser_func": "DEFAULT"
              },
              "compute_units": 10,
              "enabled": true,
              "category": {
                "deterministic": true,
                "local": false,
                "subscription": false,
                "stateful": 0
              },
              "extra_compute_units": 0
            }
          ],
          "headers": [],
          "inheritance_apis": [],
          "parse_directives": [],
          "verifications": [
            {
              "name": "",
              "values": [
                {
                "expected_value": ""
                }
              ]
            }
          ],
          "extensions": [
            {
              "name": "",
              "cu_multiplier": "",
              "rule": ""
            }
          ]
        }
      ]
    }
  },
  "deposit": "0denom"
}
```

📖 Section Reference

Each section details specific fields with descriptions and examples.

Proposal (`proposal`) 📜

Field	Description	Example
`title`	Title of the proposal.	`Add Specs: Solana`
`description`	Brief description about the purpose of the proposal.	`Adding new specification support for relaying Solana data on Lava`

Specifications (`specs`) 📘

Field	Description	Example
`index`	A unique identifier for the spec.	`JUN1`
`name`	A human-readable name for the spec.	`juno mainnet`
`enabled`	Indicates if the spec is active.	`true`
`imports`	An array of other spec indices. Allows one spec to inherit settings from another.	`["COSMOSSDKFULL"]`
`reliability_threshold`	A system parameter for data reliability.	`268435455`
`data_reliability_enabled`	Flag indicating if data reliability is enabled.	`true`
`block_distance_for_finalized_data`	The number of blocks considered safe from chain reorganizations.	`0`
`blocks_in_finalization_proof`	Number of blocks in the finality proof.	`1`
`average_block_time`	The average time (in ms) taken for a block to be produced.	`6500`
`allowed_block_lag_for_qos_sync`	Number of blocks a quality of service sync can lag by.	`2`

`min_stake_provider`	Minimum amount a provider needs to	`{"denom": "ulav`
		`"amount":`

Field	Description	Example
	stake to offer services.	`"50000000000"}`
`min_stake_client`	*(deprecated)* Minimum amount a client needs to stake to access services.	`{"denom": "ulava", "amount": "5000000000"}`

API Collections (`api_collections`) 🗂️

Field	Description	Example
`enabled`	Indicates if the API collection is active.	`true`
`collection_data`	Contains data related to the collection.	`{"api_interface": "rest", "internal_path": "", "type": "GET", "add_on": ""}`
`apis`	An array containing details of each API in the collection.	Array of API objects
`headers`	Headers to be included in the API requests.	`[]`
`inheritance_apis`	An array of APIs inherited from imported specs.	`[]`
`parse_directives`	Directives to parse the API responses.	`[]`
`verifications`	Contains verification details.	`{"name": "chain-id", "values": [ { "expected_value": "juno-1" } ]}`

API Collection Data (`collection_data`)

Field	Description	Example
`api_interface`	Interface of the API (e.g., `rest`, `grpc`).	`rest`
`internal_path`	Internal path for the API call.	``
`type`	HTTP method for the API request.	`GET`
`add_on`	Name of add-on collection belongs to	`debug`

Service APIs (`apis`) ⚙️

Field	Description	Example
`name`	Name of the API.	`juno.mint.Query/AnnualProvisions`
`block_parsing`	Describes how block heights are derived from API requests.	`{"parser_arg": ["latest"], "parser_func": "DEFAULT"}`
`compute_units`	Number of compute units required for the API.	`10`
`enabled`	Indicates if the API is active.	`true`
`category`	Specifies the category of the API.	`{"deterministic": true, "local": false, "subscription": false, "stateful": 0}`
`extra_compute_units`	Additional compute units if required.	`0`

Block Parsing(`block_parsing`)

Details on how block heights are derived from API requests.

Field	Description	Example
`parser_arg`	Arguments for the parser function.	`["latest"]`
`parser_func`	The function used for parsing.	`DEFAULT`

Service API Categories(`category`)

Field	Description	Example
`deterministic`	Indicates if the API's outcome is deterministic.	`true`
`local`	Specifies if the API call is local.	`false`
`subscription`	Indicates if the API supports subscription.	`false`
`stateful`	Describes the statefulness of the API. A value of `0` means it's stateless.	`0`

Verification(`verifications`)

Verification details used to validate the data.

Field	Description	Example
`name`	Name of the verification.	`chain-id`
`values`	Array containing expected values.	`[ { "expected_value": "juno-1" } ]`

Extensions (`extensions`)

Field	Description	Example
`name`	Name of the extension.	`archi`

Field	Description	Example
`cu_multiplier`	Compute units multiplier for the extension.	`5`
`rule`	Specific rules associated with the extension. (e.g., block number)	`block: 254`

Deposit (`deposit`) 💰

Represents the amount deposited by the user for the proposal.

Field	Description	Example
`deposit`	Amount deposited for the proposal in a particular denomination.	`10000000ulava`

📖 Glossary

Terms 📚

`average_block_time`

This value represents the typical duration, in milliseconds, between consecutive blocks being added to the blockchain. It's essential for quality of service (QoS) considerations, ensuring timely and efficient data relay without causing undue strain on the network or the nodes.

`allowed_block_lag_for_qos_sync`

This configuration determines how many blocks behind the latest block a provider can be before their QoS score begins to degrade. It essentially quantifies the maximum allowab "out-of-sync" state for a provider, beyond which their performance is deemed suboptim

For instance, if the network's latest block number is 1000 and a provider's latest block number is 995 with an "allowed_block_lag_for_qos_sync" of 5, their QoS score will start to be negatively impacted.

▼ ▤ `compares_hashes`

When set to true, it activates the data reliability features of the Lava network for the specified chain. This involves constantly comparing and validating block hashes from different nodes to guarantee data authenticity and prevent any malicious or erroneous data propagation.

▼ ▤ `deposit`

In a decentralized setup, actions like adding or updating specs may need consensus or approval. The "deposit" specifies the amount of "ulava" (the native token of the Lava network) that must be deposited as a proposal spec admission fee. It's akin to a security deposit or stake, ensuring that only serious and genuine proposals are submitted, and potentially safeguarding against spam or malicious actions.

▼ ▤ `finalization_criteria`

This parameter addresses the issue of blockchain finality. In the context of blockchains, particularly Proof-of-Work chains like Ethereum, blocks can sometimes be "orphaned" due to network forks. The "finalization_criteria" value represents the number of blocks back from the current block number that we deem "finalized" or irreversible.

For instance, with a "finalization_criteria" of 7, if the latest block number is 1000, blocks 993 and earlier are considered finalized. By doing so, the system safeguards against relaying data from blocks that might later get rejected or orphaned.

▼ ▤ `reliability_threshold`

This threshold determines the frequency at which free data reliability messages are broadcasted. At its essence, it dictates how resilient and trustworthy the data relayed is.

threshold is represented in hexadecimal format and functions as a mask to determine the frequency of reliability messages:

- **0x0FFFFFFF**: This implies that roughly 1 out of every 16 messages is a data reliability message. It's relatively infrequent, optimizing for efficiency over reliability.
- **0x8FFFFFFF**: Indicates a higher frequency – about 1 reliability message for every 2 standard messages. This is a middle-ground setting, balancing both efficiency and reliability.
- **0xFFFFFFFF**: The maximum setting where every message is a data reliability message. It prioritizes reliability above all, ensuring that data integrity is maintained at all times.

▾ ▤ `saved_blocks`

It corresponds to the number of previously finalized blocks (as determined by "finalization_criteria") that providers should retain and attach to their responses for enhanced reliability. By providing a history of previous blocks, it ensures data consistency and allows for cross-validation of data among different providers.

Parsing 🧩

Parsing is a critical aspect when interacting with diverse chains, as each chain returns data in a different format. The Lava Network has established parsing protocols to handle these variations effectively.

▾ Parsing Functions

The parsing functions define how the returned data is processed to extract the necessary information.

- **EMPTY:** Description: The data is returned as it is without any parsing.
- **PARSE_BY_ARG:** Description: Assumes the returned data is an array. It takes an index as an argument and returns the element at that index in the returned data.

- **PARSE_CANONICAL:** Description: Assumes the returned data is a canonically structured JSON. It receives key values as an argument and progresses through the JSON structure using the keys to fetch the desired element.
- **PARSE_DICTIONARY:** Description: Assumes the returned data is a string with a key-value structure (such as KEY=VAL). It receives a key and separator as arguments and returns the value corresponding to the key.
- **PARSE_DICTIONARY_OR_ORDERED:** Description: It first tries the PARSE_DICTIONARY method, and if that fails, then it resorts to the PARSE_BY_ARG method.

▼ Parsing Fields

`block_parsing`:

Determines how to extract the block number associated with a request. This is essential for queries that are specific to certain block heights.

`result_parsing`:

Determines how to extract the desired data from the response. Depending on the structure of the data returned by the chain, the appropriate parsing method is chosen.

`function_tag`:

This is crucial for the Lava network's features, such as reliability, which require fetching certain data from the chain, like the latest block number or block hashes. The function_tag marks an endpoint as being suitable to fetch specific types of information. Some examples include getBlockNumber and getBlockByNumber.

`function_template`:

For endpoints with a defined function_tag, this template serves as a format string. It can be used by relayers to construct a query to an external chain. This ensures standardized queries across different relayers.

✏ Edit this page