# Specifications (Specs)

## Overview

Specifications (specs) are the foundational blueprints for Lava's multichain support. Presented in JSON format, they detail the bare minimum requirements for an API to run on Lava. Through these specs, Lava determines which chains and methods are supported and enabled, as well as establishes the costs, requirements, and verifications for them.

Whenever the ecosystem demands a new API, a new spec is integrated. This dynamic approach incorporates modularity and extensibility directly into the protocol and ensures Lava remains current and adaptable.

Add a Spec Learn how to propose a new chain/API on Lava
Deep Dive into Specs See a living reference manual for all the fields in a Spec

## Key Concepts

### Index

Anindex is the name of the spec. Any time a spec is referenced it will be through itsindex .

Example You can see theEVMOS spec live in production:

{ "proposal" :

{ "title" :

"Add Specs: Evmos" , "description" :

"Adding new specification support for relaying Evmos data on Lava" , "specs" :

[ { "index" :

"EVMOS" , "name" :

"evmos mainnet" , "enabled" :

true ,

### Imports ⬇

imports are references to borrowed functions from other sources. APIs/Chains of the same or similar architecture can import methods from an existing spec using itsindex and implementing any new logic.imports improve efficiency by eliminating the need to repeatedly integrate identical APIs.

Example The following spec implements both Cosmos and Ethereum methods:

"imports" :

[ "COSMOSSDK" , "ETH1" ]

### API Collection

A specification always contains anapi_collection . Theapi_collection contains the list of available methods or APIs that are activated and their respectiveapi_interfaces (e.g."rest" ,"grpc" ,"jsonrpc" ,"tendermintrpc" , etc.). In other words, it outlines all the APIs or methods that must be active and operational to support a specific chain/API. Each method listed here must be served by providers and answerable to consumers.

Example "api_collections" :

[ { "enabled" :

true , "collection_data" :

{ "api_interface" :

"rest" , "internal_path" :

"" , "type" :

"GET" , "add_on" :

"" } , "apis" :

[ { } ]

## Compute Units (CU)

Every API call has a computational overhead. To quantify this, Lava employs"compute_units" or CUs. They act as a metric, assigning a nominal "cost" to each API call. This ensures transparent resource allocation and utilization, allowing consumers to gauge the intensity of their calls, and providers, in turn, to be rewarded based upon the intensity of compute.

Example "apis" :

[ { "name" :

"/evmos/claims/v1/claims_records" , "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 }

## Add-Ons

Add-Ons ("add-on" ) introduce optional new methods and APIs which are beyond the basic requirements for a chain/API. They are akin to plugins or modules that specific consumers may time-to-time request and providers may choose to serve for additional rewards. This allows for supplementary functionalities to be outlined inside a spec, giving both providers and consumers flexibility in customizing their experiences.

Example The following is a snippet of thedebug add-on for ourETH1 spec:

"collection_data" :

{ "api_interface" :

"jsonrpc" , "internal_path" :

"" , "type" :

"POST" , "add_on" :

"debug" } , "apis" :

[ { "name" :

"debug_getBadBlocks" , "block_parsing" :

{ "parser_arg" :

[ "latest" ] , "parser_func" :

"DEFAULT" } ,

## Extensions 〰

"extensions" allow for the adjustment or enhancement of existing methods and APIs within a spec for special use cases. They provide the means to tweak, optimize, or expand current functions based on the needs of a subset of consumers who require more functionality from specified method calls. This allows for alternative functionalities to be outlined inside a spec, giving both providers and consumers flexibility to serve / request special functions.

Example The following is a snippet of the"archive" extension from ourETH1 spec:

"extensions" :

[ { "name" :

"archive" , "cu_multiplier" :

5 , "rule" :

{ "block" : 254 } } ] This example specifies archive nodes who receive a "cu_multiplier " (hence more rewards) for returning earlier blocks.

# Examples

## Blockchains

- Lava
- Ethereum
- Axelar
- Evmos

## Rich APIs

- IBC
- CosmWasm
- Web3 P2P DNS Resolution (Outdated)