Boing Network — JSON-RPC API Specification

> ***Version:*** *0.1*
> ***Transport:*** *HTTP POST*
> ***Encoding:*** *JSON-RPC 2.0*
> ***References:*** *RUNBOOK.md*

Overview

Boing nodes expose a JSON-RPC HTTP interface for submitting transactions, querying chain state, and simulation. Rate limiting applies per `RateLimitConfig` (see SECURITY-STANDARDS.md). When enabled, **one HTTP `POST /` consumes one token** even if the body is a **JSON-RPC batch** (array of requests). Responses use HTTP **429** with header `Retry-After: 1` (second) and a JSON-RPC error body (`-32016`) so clients can backoff without parsing the body. The same JSON body may include a sibling `boing_http` object (`code` , `message` , `request_id`) for log correlation.

Diagnosing `-32601` (method not found)

If `boing_chainHeight` works but newer methods (e.g. `boing_getSyncState` , `boing_getNetworkInfo` , `boing_getLogs` , `boing_getTransactionReceipt`) return `-32601` **Method not found**, the HTTP endpoint is **not** running the same RPC surface as **current** `boing-node` in this repository: the process is usually an **older binary** or traffic is going through a **gateway that only whitelists some method names**. Rebuild and restart the node from this repo (`cargo build -p boing-node --release` , see RUNBOOK.md), or point the client at the node's JSON-RPC port with no filtering.

Use `boing-sdk` `probeBoingRpcCapabilities` / `npm run probe-rpc` (from `boing-sdk` after `npm run build` , or from the repo root) — the printed JSON includes `clientVersion` / `supportedMethodCount` when `boing_clientVersion` / `boing_rpcSupportedMethods` exist; the `diagnosis` field explains gaps when `explainBoingRpcProbeGaps` applies. For a single pass that also checks `GET /openapi.json` , `/.well-known/boing-rpc` , and `/live.json` , run `npm run boing:smoke` from the repo root (after `npm run build` in `boing-sdk`) or call `doctorBoingRpcEnvironment` in code.

Base URL

```
http://<host>:<rpc_port>/
```

Default RPC port: `8545` .

The same HTTP listener serves `GET /live` , `GET /ready` , `GET /ws` , `GET /openapi.json` , `GET /.well-known/boing-rpc` , and `POST /` (JSON-RPC). `GET /` (root) returns **405 Method Not Allowed** with `Allow: GET, HEAD, POST, OPTIONS` and a short plain-text hint so accidental browser or `curl` visits fail clearly. `HEAD /` returns the same **405** and `Allow` with an empty body. `OPTIONS /` advertises the same allowed methods. The in-tree handler returns **204** with no body; the **CORS** middleware may respond to browser preflights with **200** before the route runs—either way, integrators can rely on `Allow` for discovery.

JSON probes: `GET /live.json` (or `GET /live` with `Accept: application/json`) returns `{ "ok": true }` . `GET /ready.json` (or `Accept: application/json` on `/ready`) returns `{ "ready": true }` , or `503` with JSON `{ "ready": false, "reason": "peers_below_min", "peers", "min_peers" }` when `BOING_RPC_READY_MIN_PEERS` is set and peers are below the threshold (`Retry-After: 5` on **503**). `HEAD /live` and `HEAD /ready` return status only.

HTTP OpenAPI: `GET /openapi.json` returns the same OpenAPI 3.1 object as `boing_getRpcOpenApi`. `GET /.well-known/boing-rpc` returns a small JSON document with `schema_version` and path hints (no JSON-RPC round-trip required).

Request correlation: Clients may send `X-Request-Id` on any request; the node echoes it on the response. If omitted, the node assigns a UUID. Browsers reading this header from cross-origin responses need CORS `Access-Control-Expose-Headers` (the node exposes `x-request-id`, `retry-after`, and `x-boing-rpc-version`).

Response headers: Successful JSON-RPC responses include `X-Boing-RPC-version` (e.g. `boing-node/0.1.0`) so proxies and clients can detect binary skew without an extra `boing_clientVersion` call. The same header is set on `/live` and `/ready`.

Request body limit: `POST /` JSON bodies default to **8 MiB** max (large signed-transaction hex). Set `BOING_RPC_MAX_BODY_MB` (integer ≥ **1**) to override. Oversized bodies receive HTTP **413** with `Content-Type: application/json` and body `{ "code": "payload_too_large", "message", "request_id" }` when `x-request-id` is available on the response.

JSON-RPC batch: The body may be a **JSON array** of request objects. The HTTP response is a **JSON array** of response objects in the same order. Requests **without an** `id` are treated as **notifications** and produce **no entry** in the batch response. If **every** entry in the batch is a notification, the server returns **HTTP 204** with **no body** (per JSON-RPC 2.0). A **single** notification (non-batch) also returns **204**. An empty batch array `[]` returns `200` with body `[]`. Max batch length defaults to **32** and is capped at **256**; set `BOING_RPC_MAX_BATCH` to override (invalid values fall back to the default).

Browser CORS

`boing-node` sends **CORS** headers for a built-in allowlist (Boing web apps and common local dev ports). To allow **additional** browser origins without recompiling, set environment variable `BOING_RPC_CORS_EXTRA_ORIGINS` to a comma-separated list (e.g. `http://localhost:9999,https://preview.example.com`). Invalid entries are skipped with a warning in node logs.

Chain tip, committed height, and finality

Boing uses a **HotStuff-style BFT** consensus layer (`boing-consensus`): a block is appended to this node's `ChainState` only after it is **committed** through that path. There is **no separate JSON-RPC "unsafe head"** ahead of commit in the current node implementation.

Term	Meaning in this codebase
`boing_chainHeight` / `head_height` in `boing_getSyncState`	Height of the latest **committed** block stored on this node (the chain tip).
Finalized (protocol sense)	Under standard BFT assumptions (honest quorum, etc.), a committed block is not reverted by consensus.
`finalized_height` in `boing_getSyncState`	Today **equal to** `head_height`, because the node only exposes committed blocks. Reserved so a future release can report lag (e.g. optimistic execution vs commit) without breaking clients.

Syncing / lagging peers	A node that is still catching up may report a **lower** height than the rest of the network until it imports commits. Clients should not treat "height stopped increasing" as finality across the whole network without comparing multiple sources.

Wallets and observers that need a single number for "how deep is my tx" can use `boing_chainHeight` or `boing_getSyncState`; until multiple heights diverge in a future version, they are equivalent for "committed tip."

dApp network discovery

`boing_getNetworkInfo` (params `[]`) returns a single JSON object for wallets and dApps: committed tip fields (same as `boing_getSyncState`), target block time, `client_version`, optional `chain_id` / `chain_name` from environment (see below), consensus summary (`validator_count`, `model`), `native_currency`, `chain_native` (chain-wide sums over the committed account table), `developer` (doc URLs, npm package id, WebSocket `/ws` handshake, and API discovery method names), and an explicit `rpc.not_available` list.

`chain_native`: `account_count`, `total_balance`, `total_stake`, and `total_native_held` are **u128 decimal strings** (same style as `boing_getAccount`). They are the **sum** of per-account `balance` / `stake` in this node's committed state — **not** "circulating supply", "total minted", or treasury definitions. `as_of_height` is the committed tip height those sums match.

Important: Public JSON-RPC still does **not** expose **staking APY** or protocol-defined **network-wide supply** (mint/burn/treasury). The node lists remaining gaps under `rpc.not_available` (today `staking_apy`) and `rpc.not_available_note`. **Per-account** balance and stake remain available via `boing_getAccount` (and `boing_getBalance` for balance only).

Operator environment (optional):

Variable	Effect on `boing_getNetworkInfo`
`BOING_CHAIN_ID`	If set to a decimal integer (e.g. `6913` for public testnet), included as JSON number `chain_id`. If unset or invalid, `chain_id` is `null` — clients must use their configured chain id (e.g. from app env).
`BOING_CHAIN_NAME`	Optional human-readable `chain_name` (e.g. `Boing Testnet`). Omitted when unset.
`BOING_CHAIN_DISPLAY_NAME`	Optional wallet-facing display string in `end_user.chain_display_name` (does not replace `chain_name`).
`BOING_EXPLORER_URL`	Optional block explorer base URL in `end_user.explorer_url`.
`BOING_FAUCET_URL`	Optional faucet URL in `end_user.faucet_url`.

`rpc_surface` on `boing_getNetworkInfo`: mirrors `boing_health.rpc_surface` (batch cap, body limit, `boing_getLogs` bounds, WS cap, rate limit, ready peer gate).

Request Format

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "boing_chainHeight",
```

```
  "params": []
}
```

Response Format

Success:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": 42
}
```

Error:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "error": {
    "code": -32602,
    "message": "Invalid params: expected [hex_signed_tx]"
  }
}
```

Methods

Method index (`boing_rpcSupportedMethods`)

Current `boing-node` implements these JSON-RPC methods (same set returned by **`boing_rpcSupportedMethods`**, sorted alphabetically):

`boing_chainHeight`, `boing_clientVersion`, `boing_faucetRequest`, `boing_getAccount`, `boing_getAccountProof`, `boing_getBalance`, `boing_getBlockByHash`, `boing_getBlockByHeight`, `boing_getContractStorage`, `boing_getLogs`, `boing_getNetworkInfo`, `boing_getQaRegistry`, `boing_getRpcMethodCatalog`, `boing_getRpcOpenApi`, `boing_getSyncState`, `boing_getTransactionReceipt`, `boing_health`, `boing_operatorApplyQaPolicy`, `boing_qaCheck`, `boing_qaPoolConfig`, `boing_qaPoolList`, `boing_qaPoolVote`, `boing_registerDappMetrics`, `boing_rpcSupportedMethods`, `boing_simulateTransaction`, `boing_submitIntent`, `boing_submitTransaction`, `boing_verifyAccountProof`.

Implementation: `BOING_RPC_SUPPORTED_METHODS` in `crates/boing-node/src/rpc.rs` — keep this list, the router match arms, and this spec in sync when adding a method.

---

boing_submitTransaction

Submit a signed transaction to the mempool.

Field	Type	Description
Params	`[hex_signed_tx]`	Hex-encoded bincode-serialized SignedTransaction

Example:

```
{"jsonrpc":"2.0","id":1,"method":"boing_submitTransaction","params":["0x..."]}
```

P2P: When `--p2p-listen` is set and the tx is **admitted** to the mempool, the node also **gossips** the same `SignedTransaction` on libp2p topic `boing/transactions` so peers can verify and enqueue it locally (TECHNICAL-SPECIFICATION.md §12.3, RUNBOOK.md §8.1).

Result (success): `{ "tx_hash": "ok" }` — current `boing-node` uses the literal `"ok"` when the tx is accepted into the mempool (not a 32-byte transaction id). Track inclusion via `boing_getBlockByHeight` / receipts / account nonce, or compute the signable `tx_id` client-side from the signed payload if your tooling needs a stable hex.

QA pool (Unsure): If mempool QA returns Unsure **and** governance allows the pool to accept work (`qa_pool_config` : non-zero `max_pending_items` and either `administrators` or `dev_open_voting`), the node enqueues the deployment and responds with `-32051` and `data: { "tx_hash": "0x..." }` . **Governance-listed administrators** vote via `boing_qaPoolVote` . Hard caps (`max_pending_items` , `max_pending_per_deployer`) prevent pool congestion; when full, `-32055` / `-32056` apply instead of enqueueing.

Operator RPC (optional): When the node process has environment variable `BOING_OPERATOR_RPC_TOKEN` set to a non-empty string, `boing_qaPoolVote` and `boing_operatorApplyQaPolicy` require HTTP header `X-Boing-Operator: <same token>` . If the variable is unset, behavior matches earlier releases (no header check). Use this on any RPC endpoint reachable from untrusted networks so pool votes cannot be triggered by spoofing an admin hex alone.

Does the pool need RPC to "run"? No. The node **owns** the pool: when QA returns Unsure and governance allows it, enqueueing happens inside normal transaction/mempool handling (`boing_submitTransaction` may return `-32051`). No operator client is required for items to enter the queue or for the node to age them out per config. JSON-RPC is how **operators** *inspect and change* the pool—** `boing_qaPoolList` **, `boing_qaPoolConfig` , `boing_qaPoolVote` , `boing_operatorApplyQaPolicy` . For routine governance work, the **Boing Network desktop hub** (QA operator view) calls those methods over HTTP, so a terminal or `boing` **CLI** is optional (CLI remains useful for scripts and file-based `boing qa apply`).

---

boing_qaPoolList

List pending items in the community QA pool (same `tx_hash` keys as `-32051`).

Field	Type	Description
Params	`[]`	None
Result	`{ items: [...] }`	Each item: `tx_hash`, `bytecode_hash`, `deployer` (hex), `allow_votes`, `reject_votes`, `age_secs`.

---

boing_qaPoolVote

Cast a vote on a pending pool item. When quorum and allow/reject thresholds are met, the item is resolved; on **Allow**, the stored signed transaction is inserted into the mempool.

Field	Type	Description

Params	`[tx_hash_hex, voter_hex, vote]`	`vote` is `allow`, `reject`, or `abstain` (case-insensitive). Only accounts listed in governance `qa_pool_config.administrators` may vote, unless `dev_open_voting` is true with an empty admin list (local dev).
Result	`{ outcome: "pending" | "reject" | "allow", mempool?: boolean, duplicate?: boolean, error?: string }`	On `allow` with `mempool: true`, the tx is in the mempool.

Errors: `-32052` no pending item for `tx_hash`; `-32053` voter is not a governance QA administrator; **`-32057`** operator authentication required (see **Operator RPC** above).

---

boing_operatorApplyQaPolicy

Replace the node's in-memory QA registry and pool governance config (same effect as loading `qa_registry.json` / `qa_pool_config.json` at startup, plus persistence to the node data directory). Intended for operators; requires **`X-Boing-Operator`** when **`BOING_OPERATOR_RPC_TOKEN`** is set.

Field	Type	Description
Params	`[qa_registry_json, qa_pool_config_json]`	Two **strings**, each the full JSON document (not a path).
Result	`{ ok: true }`	Policy applied.

Errors: **`-32057`** missing or wrong operator header when the token is configured; **`-32602`** invalid JSON or schema.

CLI: `boing qa apply --registry <path> --pool <path> [--operator-token …]` (also reads `BOING_OPERATOR_RPC_TOKEN`).

---

boing_qaPoolConfig

Read effective QA pool governance parameters and current queue depth (no params).

Field	Type	Description
Params	`[]`	None
Result	object	`max_pending_items`, `max_pending_per_deployer`, `review_window_secs`, quorum/threshold fractions, `default_on_expiry`, `dev_open_voting`, `administrator_count`, `accepts_new_pending`, `pending_count`.

---

boing_getQaRegistry

Return the **effective protocol QA rule registry** the node uses for deployment checks (read-only, no authentication). Same JSON shape as on-disk **`qa_registry.json`** and as the first argument to **`boing_operatorApplyQaPolicy`**.

Field	Type	Description
Params	`[]`	None

Result	object	`max_bytecode_size`, `blocklist` (array of 32-byte arrays), `scam_patterns`, `always_review_categories` (array of strings), `content_blocklist`.

Reference: Canonical baseline JSON for comparison lives in the monorepo at **`docs/config/qa_registry.canonical.json`** — see **`docs/config/CANONICAL-QA-REGISTRY.md`**. Live nodes may differ after governance updates.

---

boing_clientVersion

Field	Type	Description
Params	`[]`	None

Result: string — e.g. `boing-node/0.1.0` (crate version). Lets clients and **`npm run probe-rpc`** confirm they are talking to a freshly built binary.

---

boing_rpcSupportedMethods

Field	Type	Description
Params	`[]`	None

Result: JSON array of strings — sorted list of **`boing_*`** method names implemented by this process (including **`boing_clientVersion`** and **`boing_rpcSupportedMethods`**). Intended for discovery and debugging; keep in sync with the node's RPC router when adding methods.

---

boing_chainHeight

Return the height of the latest **committed** block on this node (see Chain tip, committed height, and finality).

Field	Type	Description
Params	`[]`	None

Result: `u64`

---

boing_health

Lightweight **liveness** and **build identity** for probes, dashboards, and load balancers. Single round-trip; includes committed **`head_height`** and the same optional **`chain_id`** / **`chain_name`** semantics as **`boing_getNetworkInfo`** (from **`BOING_CHAIN_ID`** / **`BOING_CHAIN_NAME`** on the node process).

Field	Type	Description
Params	`[]`	None

Result:

```
{
  "ok": true,
  "client_version": "boing-node/0.1.0",
```

```
  "chain_id": 6913,
  "chain_name": "Boing Testnet",
  "head_height": 42,
  "rpc_surface": {
    "jsonrpc_batch_max": 32,
    "websocket_max_connections": 0,
    "http_rate_limit_requests_per_sec": 100,
    "ready_min_peers": null,
    "http_max_body_megabytes": 8,
    "get_logs_max_block_range": 128,
    "get_logs_max_results": 2048,
    "max_log_topic_filters": 4
  },
  "rpc_metrics": {
    "rate_limited_total": 0,
    "json_parse_errors_total": 0,
    "batch_too_large_total": 0,
    "method_not_found_total": 0,
    "websocket_cap_rejects_total": 0
  }
}
```


- **`chain_id` / `chain_name`:** `null` when the corresponding env var is unset or invalid (same as **`boing_getNetworkInfo`**).
- **`rpc_surface`:** Operator-facing snapshot of limits (same sources as env vars **`BOING_RPC_MAX_BATCH`**, **`BOING_RPC_WS_MAX_CONNECTIONS`**, **`BOING_RPC_MAX_BODY_MB`**, node **`RateLimitConfig.requests_per_sec`**, **`BOING_RPC_READY_MIN_PEERS`**, plus fixed **`boing_getLogs`** bounds). **`ready_min_peers`** is **`null`** when no minimum is configured.
- **`rpc_metrics`:** Cumulative counters since process start (best-effort diagnostics).

---

boing_getSyncState

Structured view of the node's committed chain tip for clients that want explicit **head** vs **finalized** fields. Today both heights are identical.

Field	Type	Description
Params	`[]`	None

Result:


```
{
  "head_height": 42,
  "finalized_height": 42,
  "latest_block_hash": "0x..."
}
```


- **`head_height`:** Same value as `boing_chainHeight`.
- **`finalized_height`:** Same as `head_height` in the current implementation; may diverge in a future release if the node exposes pre-commit or optimistic data.

- **`latest_block_hash`:** BLAKE3 hash of the tip block's header (`Block::hash()`), 32-byte hex with `0x` prefix.

---

boing_getNetworkInfo

Single-call snapshot for dApps: chain metadata (from env when configured), tip state, timing, client build, consensus summary, **`chain_native`** aggregates, and **`rpc.not_available`** (see dApp network discovery).

Field	Type	Description
Params	`[]`	None

Result (shape):

```
{
  "chain_id": 6913,
  "chain_name": "Boing Testnet",
  "head_height": 42,
  "finalized_height": 42,
  "latest_block_hash": "0x...",
  "target_block_time_secs": 2,
  "client_version": "boing-node/0.1.0",
  "consensus": {
    "validator_count": 4,
    "model": "hotstuff_bft"
  },
  "native_currency": {
    "symbol": "BOING",
    "decimals": 18
  },
  "chain_native": {
    "account_count": 42,
    "total_balance": "…",
    "total_stake": "…",
    "total_native_held": "…",
    "as_of_height": 42
  },
  "developer": {
    "repository_url": "https://github.com/chiku524/boing.network",
    "rpc_spec_url": "https://github.com/chiku524/boing.network/blob/main/docs/RPC-API-
SPEC.md",
    "dapp_integration_doc_url":
"https://github.com/chiku524/boing.network/blob/main/docs/BOING-DAPP-INTEGRATION.md",
    "sdk_npm_package": "boing-sdk",
    "websocket": {
      "path": "/ws",
      "handshake": { "type": "subscribe", "channel": "newHeads" },
      "event_types": ["newHead"]
    },
    "api_discovery_methods": [
      "boing_getRpcMethodCatalog",
      "boing_getRpcOpenApi",
```

```
      "boing_rpcSupportedMethods"
    ],
    "http": {
      "live_path": "/live",
      "ready_path": "/ready",
      "jsonrpc_post_path": "/",
      "response_header_rpc_version": "x-boing-rpc-version"
    }
  },
  "rpc": {
    "not_available": ["staking_apy"],
    "not_available_note": "…"
  }
}
```

- **`chain_id` / `chain_name`:** From **`BOING_CHAIN_ID`** / **`BOING_CHAIN_NAME`** when set on the node process; **`chain_id`** is **`null`** when not configured.
- **`validator_count`:** Size of this node's in-process validator set (HotStuff BFT); not a network-wide discovery endpoint for all operators.
- **`native_currency.decimals`:** Display hint for integrators; balances on **`boing_getBalance`** / **`boing_getAccount`** are still whole-unit u128 strings per those methods.
- **`chain_native`:** Sums over this node's committed account table; see dApp network discovery.
- **`developer`:** Stable links and discovery hints for integrators. URLs default to the public GitHub tree; operators may override with environment variables (see table below).

Variable	Effect
`BOING_DEVELOPER_REPOSITORY_URL`	Base repository URL used to derive default doc links when spec/dapp URLs are unset.
`BOING_DEVELOPER_RPC_SPEC_URL`	Overrides `developer.rpc_spec_url`.
`BOING_DEVELOPER_DAPP_DOC_URL`	Overrides `developer.dapp_integration_doc_url`.

HTTP probes `GET /live` and `GET /ready`

Plain-text endpoints for orchestrators (Kubernetes **`livenessProbe`** / **`readinessProbe`**, Docker **`HEALTHCHECK`**, load balancers) without parsing JSON-RPC.

Path	Meaning
`GET /live`	Process is running and the HTTP server is accepting connections. Does **not** read chain state.
`GET /ready`	Node can acquire a read lock on in-memory state (RPC is not wedged on the state mutex). Optionally also requires a minimum **P2P peer count** when `BOING_RPC_READY_MIN_PEERS` is set (see below).

Successful responses use HTTP **200** with a short body (`ok` / `ready`). If the peer requirement is not met, **`GET /ready`** returns **503** with a short `not_ready: peers …` message. They receive the same **`x-boing-rpc-version`** response header as **`POST /`**.

Request body limit (JSON-RPC `POST /`)

Large signed-transaction hex payloads (e.g. contract deploy) need a generous HTTP body limit. Default maximum request body size is **8 MiB**.

Variable	Effect
`BOING_RPC_MAX_BODY_MB`	Maximum JSON-RPC POST body size in **mebibytes** (integer, **≥ 1**). Invalid or missing values fall back to **8**.
`BOING_RPC_MAX_BATCH`	Max objects per JSON-RPC **batch** array (default **32**, hard cap **256**). `0` is invalid and falls back to the default.
`BOING_RPC_READY_MIN_PEERS`	When set to a positive integer, `GET /ready` returns **503** unless the node reports at least that many connected P2P peers. Unset or `0`: no peer check (read lock only).
`BOING_RPC_WS_MAX_CONNECTIONS`	Max concurrent `GET /ws` WebSocket connections (**0** = unlimited). When exceeded, new handshakes receive **503**.

WebSocket `GET /ws` — newHeads

Browser-friendly push when the **local** committed tip changes (block produced or imported). Uses the same CORS allowlist as `POST /` plus `BOING_RPC_CORS_EXTRA_ORIGINS`.

1. Connect with `GET /ws` (WebSocket upgrade).
2. Send **one** text frame: `{"type":"subscribe","channel":"newHeads"}`.
3. Server replies `{"type":"subscribed","channel":"newHeads"}`.
4. On each committed tip update, server sends `{"type":"newHead","height":<u64>,"hash":"0x..."}` (BLAKE3 header hash, 32-byte hex).

Laggy subscribers may drop intermediate heads (**broadcast** channel); clients should reconcile with `boing_getSyncState` / `boing_chainHeight` if they need every height.

Public RPC operators may set `BOING_RPC_WS_MAX_CONNECTIONS` so `newHeads` cannot exhaust file descriptors or CPU from unbounded subscribers.

---

boing_getRpcMethodCatalog

Returns the embedded **method catalog** (JSON Schema–style `params` / `result` hints per method) shipped with the node binary. Intended for codegen, docs generators, and IDE tooling.

Field	Type	Description
Params	`[]`	None

Result: JSON object with `description` and `methods` (array of `{ name, summary, params, result }`).

---

boing_getRpcOpenApi

Returns a **minimal OpenAPI 3.1** JSON document describing `POST /` (JSON-RPC envelope), `GET /ws` (subscribe protocol), and plain-text probes `GET /live` / `GET /ready`. Complements the human-readable spec and `boing_getRpcMethodCatalog`.

Field	Type	Description
Params	`[]`	None

Result: OpenAPI root object (`openapi` , `info` , `paths` , `components` , …).

---

boing_getBalance

Get the spendable balance for an account. **Recommended for wallets** (e.g. boing.express) to display balance without deriving from state.

Field	Type	Description
Params	`[hex_account_id]`	32-byte AccountId (hex)
Result	`{ balance: string }`	Native BOING balance as **whole units** (u128 decimal string). Wallets and explorers should **not** assume a fixed `10^18` "wei"-style divisor unless a future token standard adds it.

Example: `{"jsonrpc":"2.0","id":1,"method":"boing_getBalance","params":["0x..."]}` → `{"jsonrpc":"2.0","id":1,"result":{"balance":"1000000"}}`

---

boing_getAccount

Get full account state (balance, nonce, stake). **Recommended for wallets** to build transactions (nonce) and show balance/stake.

Field	Type	Description
Params	`[hex_account_id]`	32-byte AccountId (hex)
Result	`{ balance: string, nonce: number, stake: string }`	balance and stake are u128 as decimal strings; nonce is u64. If account does not exist, returns balance "0", nonce 0, stake "0".

Example: `{"jsonrpc":"2.0","id":1,"method":"boing_getAccount","params":["0x..."]}` → `{"jsonrpc":"2.0","id":1,"result":{"balance":"1000000","nonce":5,"stake":"0"}}`

---

boing_getBlockByHeight

Get a block by height.

Field	Type	Description
Params	`[height]` or `[height, include_receipts]`	Block height (u64); optional boolean — when `true`, adds `receipts` array (same order as `transactions`, `null` if no receipt cached).

Result: Block object or `null` if not found. `header` includes `receipts_root` (hex-encoded 32-byte Merkle root over execution receipts; see protocol spec). Optional `hash` field (hex) is added for convenience. With `include_receipts: true` , each entry matches **`boing_getTransactionReceipt`** shape or `null` .

---

boing_getTransactionReceipt

Lookup execution outcome for an included transaction (same id as `Transaction::id()` / mempool `tx_hash`).

Field	Type	Description
Params	`[hex_tx_id]`	32-byte transaction id (hex)

Result: Receipt object or `null` if unknown (e.g. not yet included, or node predates receipt persistence).

Receipt: `{ tx_id, block_height, tx_index, success, gas_used, return_data, logs, error }` — `return_data` is hex; `logs` is an array of `{ topics: string[], data: string }` (each topic is `0x` + 32-byte hex, `data` is hex). Logs are populated for `ContractCall` and for `ContractDeploy` **only when bytecode uses init-code mode** (leading `0xFD` prefix; see `TECHNICAL-SPECIFICATION.md` §4.4). `error` is set when `success` is false.

---

boing_getLogs

Query execution logs across a **bounded** range of committed blocks. Intended for indexers and wallets; nodes enforce limits to keep scans cheap.

Field	Type	Description
Params	`[filter]`	Single object (camelCase keys below).
`fromBlock`	number or string	Start height (inclusive). JSON number or decimal / `0x` hex string.
`toBlock`	number or string	End height (inclusive). Must be `>= fromBlock`.
`address`	string (optional)	32-byte `AccountId` hex. When set, only logs attributed to that contract are returned (`ContractCall` target, or the deployed contract address for deploy txs that used **init-code** mode — leading bytecode byte `0xFD`, see `TECHNICAL-SPECIFICATION.md` §4.4).
`topics`	array (optional)	Up to 4 entries; each is `null` (wildcard) or a 32-byte topic hex string. **Positional match:** entry `i` filters `topic_i` of each log (same index slot).

Limits (reference implementation): inclusive block span at most **128**; at most **2048** log entries per call. Exceeding span returns JSON-RPC error `-32602`; exceeding the result cap returns `-32603`.

Result: JSON array of objects:

`{ block_height, tx_index, tx_id, log_index, address, topics, data }`

- `address` is hex or `null` when the node cannot attribute an emitting contract.
- Order is block height ascending, then transaction order in the block, then `log_index`.

Example: `{"jsonrpc":"2.0","id":1,"method":"boing_getLogs","params":[{"fromBlock":1,"toBlock":10,"topics":[null,"0x000…"]}]}`

---

boing_getBlockByHash

Get a block by hash.

Field	Type	Description
Params	`[hex_block_hash]` or `[hex_block_hash, include_receipts]`	Optional boolean — same as `boing_getBlockByHeight`.

Result: Block object or `null` if not found.

---

boing_getAccountProof

Get a Merkle proof for an account.

Field	Type	Description
Params	`[hex_account_id]`	32-byte AccountId (hex)

Result: `{ proof: string, root: string, value_hash: string }`

---

boing_verifyAccountProof

Verify an account Merkle proof.

Field	Type	Description
Params	`[hex_proof, hex_state_root]`	Proof and expected root (hex)

Result: `{ valid: boolean }`

---

boing_getContractStorage

Read a single 32-byte Boing VM storage word for a contract (same semantics as `SLOAD`: missing slot → zero word). Useful for indexers and wallets that know the storage key layout (e.g. reference NFT / token conventions).

Field	Type	Description
Params	`[hex_contract_id, hex_storage_key]`	Two 32-byte values (hex)

Result: `{ value: string }` — `value` is `0x` + 64 hex chars (32 bytes).

Native constant-product AMM (chain 6913 — integration note)

Wallets and dApps still **configure** the pool `AccountId` in their own env (the JSON-RPC methods are generic). For **public testnet**, operators maintain a single **canonical** pool id so tutorials and UIs can default to one known-good deployment.

Canonical public testnet pool `AccountId`	`0xffaa1290614441902ba813bf3bd8bf057624e0bd4f16160a9d32cd65d3f4d0c2` — **v1** bytecode, **CREATE2** + `NATIVE_CP_POOL_CREATE2_SALT_V1`, `purpose_category dapp`. Deployer `0xc063512f42868f1278c59a1f61ec0944785c304dbc48dec7e4c41f70f666733f`. Published **2026-04-03** (OPS-CANONICAL-TESTNET-NATIVE-AMM-POOL.md § Published).
boing.finance (separate	**Source constant:** `frontend/src/config/boingCanonicalTestnetPool.js` → `CANONICAL_BOING_TESTNET_NATIVE_CP_POOL_HEX`. **Cloudflare Pages / CI build:**

repo)	**`REACT_APP_BOING_NATIVE_AMM_POOL`** (same 64-hex `AccountId`). Also wire **`nativeConstantProductPool`** (or equivalent in `contracts.js`) for chain **6913**. Redeploy the app after any change to this hex. Optional: depend on npm **`boing-sdk`** and import **`CANONICAL_BOING_TESTNET_NATIVE_CP_POOL_HEX`** (mirror of this table — see row below) plus shared RPC helpers / `explainBoingRpcError`.
boing-sdk	**`CANONICAL_BOING_TESTNET_NATIVE_CP_POOL_HEX`** — exported from [boing-sdk/src/canonicalTestnet.ts](); convenience for tutorials and TS apps. **Authoritative** values remain **this spec** and TESTNET.md §5.3.

Need	RPC / approach
Reserve A / B	Two **`boing_getContractStorage`** calls: `hex_contract_id` = pool account; storage keys are the **32-byte big-endian layout** for the pool program (see NATIVE-AMM-CALLDATA.md and `boing_execution::reserve_a_key` / `reserve_b_key`). Each `value` word holds the u128 reserve in the **low 16 bytes** (high 16 zero), matching reference-token word style.
Swap / liquidity	Signed **`contract_call`** transactions (calldata per NATIVE-AMM-CALLDATA.md); preflight **`boing_simulateTransaction`** / **`boing_qaCheck`** on deploy bytecode as usual. **Canonical pool deploy:** prefer **`create2_salt: Some(NATIVE_CP_POOL_CREATE2_SALT_V1)`** on purpose deploy (fixed salt in `native_amm.rs`) so the pool **`AccountId`** is predictable from deployer + bytecode — see NATIVE-AMM-CALLDATA.md § CREATE2.
Automated regression	Repository test: `cargo test -p boing-node --test native_amm_rpc_happy_path` (deploy → add liquidity → swap; asserts reserves via `boing_getContractStorage`).

Publish checklist (for **future** pool rotations): OPS-CANONICAL-TESTNET-NATIVE-AMM-POOL.md. Integration checklist **A6.4** / **A1.5** are satisfied for the current canonical hex; keep **boing.finance** and **`boing-sdk`** mirrors in sync when the table above changes.

Native DEX directory + ledger router (integration note)

Multi-pair flows use **generic** JSON-RPC (`contract_call` , `getContractStorage` , `getLogs` , `simulateTransaction`) — no new method names. Specs and bytecode live in the node repo:

Artifact	Doc	Notes
Pair directory	NATIVE-DEX-FACTORY.md	`register_pair` / `pairs_count` / `get_pair_at`; count key + `Log3` indexing
Ledger router	NATIVE-DEX-LEDGER-ROUTER.md	Forwards **128-byte** pool calldata via nested `Call` — **v1/v3 pools only**
SDK	`boing-sdk`	`nativeDexFactory*`, `nativeDexLedgerRouter`, `create2` prediction helpers
Regression	`cargo test -p boing-node --test native_dex_factory_rpc_happy_path`	Deploy pool + directory → `register_pair` → `pairs_count` / `get_pair_at` receipts

---

boing_simulateTransaction

Simulate a transaction without applying it.

Field	Type	Description
Params	`[hex_signed_tx]`	Hex-encoded SignedTransaction

Result: `{ gas_used: number, success: boolean, return_data: string, logs?: array, error?: string, suggested_access_list: { read: string[], write: string[] }, access_list_covers_suggestion: boolean }` — on success, `return_data` is hex (contract return buffer) and `logs` matches receipt log shape; on failure, `return_data` is `"0x"` and `logs` is `[]`. **`suggested_access_list`** is a **heuristic** minimum account set for parallel scheduling from the tx payload (see `TECHNICAL-SPECIFICATION.md` §4.2); **`access_list_covers_suggestion`** is `true` when the signed tx's declared access list includes every account in that suggestion (read or write). Hints are included on both success and failure so clients can fix access lists before submit.

---

boing_registerDappMetrics

Register a dApp for incentive tracking.

Field	Type	Description
Params	`[hex_contract, hex_owner]`	Contract and owner AccountIds (hex)

Result: `{ registered: true, contract: string, owner: string }`

---

boing_submitIntent

Submit a signed intent for solver fulfillment.

Field	Type	Description
Params	`[hex_signed_intent]`	Hex-encoded SignedIntent

Result: `{ intent_id: string }`

---

boing_qaCheck (optional — when QA is enabled)

Pre-flight check for a deployment without submitting. Allows clients to see whether bytecode (and optional purpose declaration) would be **Allow**, **Reject**, or **Unsure** (pool) before calling `boing_submitTransaction`.

Field	Type	Description
Params	`[hex_bytecode]` or `[hex_bytecode, purpose_category, description_hash?, asset_name?, asset_symbol?]`	Bytecode only runs size/opcode/blocklist rules. With `purpose_category`, the same **full mempool QA** path applies as in `boing_submitTransaction` for deploy payloads (including optional 32-byte `description_hash`, then optional `asset_name` / `asset_symbol` for content policy). To pass only name/symbol without a real description commitment, use a placeholder 32-byte hex (e.g. all zeros).
Result	`{ result, rule_id?, message?, doc_url? }`	`result`: `"allow"`, `"reject"` (rule_id/message when applicable), or `"unsure"` (community QA pool). Mirrors mempool `boing_qa` for contract deploy.

Errors: When QA is not enabled, returns `-32601` (method not found) or a dedicated code. When QA rejects: use structured error code below.

---

boing_faucetRequest (testnet only)

Request testnet BOING for an account. Only available when the node is started with `--faucet-enable`. **Do not enable on mainnet.**

Field	Type	Description
Params	`[hex_account_id]`	32-byte account ID (hex). Recipient of the faucet transfer.

Result: `{ ok: true, amount: number, to: string, message: string }`

Rate limit: 1 request per 60 seconds per account ID. Returns `-32016` with message "Faucet cooldown" if called too soon.

P2P: Same as **`boing_submitTransaction`** — when **`--p2p-listen`** is set and the built transfer is admitted, the signed faucet tx is **gossiped** on **`boing/transactions`**.

Errors: `-32601` Faucet not enabled; `-32000` Faucet account not initialized or balance too low.

Error Codes

Code	Meaning
-32600	Invalid Request
-32601	Method not found
-32602	Invalid params
-32000	Server error
-32016	Rate limit exceeded
-32050	**QA: Deployment rejected** — Transaction rejected by protocol QA (e.g. bytecode or purpose rule). Response SHOULD include `data: { rule_id: string, message: string }` for structured feedback. See QUALITY-ASSURANCE-NETWORK.md.
-32051	**QA: Pending pool** — Deployment referred to governance QA pool (result: Unsure). Response includes `data: { tx_hash: string }` (hex).
-32052	**QA pool** — No pending item for the given `tx_hash`.
-32053	**QA pool** — Voter is not a governance QA administrator.
-32054	**QA pool disabled** — Governance has not enabled the pool (e.g. no `administrators` and `dev_open_voting` false, or `max_pending_items` is 0).
-32055	**QA pool full** — Global `max_pending_items` reached; optional `data.reason: "pool_full"`.
-32056	**QA pool deployer cap** — Sender exceeded `max_pending_per_deployer`; optional `data.reason: "deployer_cap"`.

-32057	**Operator RPC auth** — `boing_qaPoolVote` or `boing_operatorApplyQaPolicy` called without valid `X-Boing-Operator` while `BOING_OPERATOR_RPC_TOKEN` is set on the node.

Boing Network — Authentic. Decentralized. Optimal. Sustainable.