JANZEER BLOCKCHAIN

REST API Reference

Endpoints, envelopes & examples

Document v1.0 · Janzeer Network · 2026

Janzeer REST API Reference

This is the complete, lookup-oriented reference for the **Janzeer** node's HTTP API: every resource, method, path, parameter, request body, and returned payload. Janzeer is a fixed-supply, permissionless **stake-free** BFT Layer-1 with a non-custodial HD-wallet crypto scheme — the node verifies signatures and applies state transitions but **never holds keys**, so there are no key-generation, seed, or signing endpoints here. For a tutorial-style walkthrough (running a node, the client-side signing pipeline, parity vectors) see the Developer Guide; for exact byte layouts and constants see the Yellow Paper. This document is a reference, not a tutorial.

> **Client SDKs.** Official typed clients exist for TypeScript (`janzeer-sdk`), Dart (`janzeer_sdk`) and Kotlin/JVM/Android (`org.janzeer:janzeer-sdk-client`); they wrap every endpoint below plus JSON-RPC and the WebSocket feeds — see the Developer Guide §5. Hand-written clients should read §1.3 (errors are enveloped) carefully.

---

1. Conventions

1.1 Base URL and port

Item	Value
Base path	`/api/v1/`
REST port	`7019` (`server.port`)
Example host	`http://localhost:7019`
Content type	`application/json` (request and response)

All endpoint paths in this document are relative to `http://<host>:7019`. Unless a method is stated explicitly, every endpoint is `GET`.

1.2 Response envelope

Every response under `/api/v1/**` is wrapped in a standard envelope; the actual result is always in `payload`:

```
{
  "timestamp": 1719700000000,
  "version": "1.1.0",
  "payload": { }
}
```

Field	Type	Meaning
`timestamp`	`Long`	server time (epoch milliseconds) when the response was produced
`version`	`String`	envelope version, currently `"1.1.0"`
`payload`	`T`	the endpoint-specific result (object, array, scalar, or `null`)

Paths that are NOT wrapped (returned raw, no envelope):

- `/docs/**` (Swagger UI and the OpenAPI document)
- `/swagger-resources`
- `/actuator/**` (Spring Boot Actuator)

1.3 Error shape and status codes

Errors use the same envelope as every other response; the error object is the `payload`:

```
{
  "timestamp": 1757800000000,
  "version": "1.1.0",
  "payload": { "status": 400, "message": "Invalid nonce for 0x06e1…: expected 4, got 7",
"type": "INVALID_NONCE" }
}
```

Field	Type	Meaning
`status`	`Int`	HTTP status code, echoed in the body
`message`	`String`	human-readable description
`type`	`String?`	machine-readable rejection type for transaction validation failures: `INCORRECT_HASH`, `INCORRECT_ADDRESS`, `INCORRECT_SIGNATURE`, `INCORRECT_PROMOTER_KEY`, `ALREADY_PROMOTER`, `INSUFFICIENT_BALANCE`, `INSUFFICIENT_ACTUAL_BALANCE`, `INVALID_NONCE` (message carries `expected N, got M`); absent for other errors

Two exceptions: request-body validation failures (missing/invalid fields) return `payload` as an **array** of `{message}` objects, and the rate limiter's `429`/`413` bodies are written before the envelope layer, so they are flat `{status, message}`. The official SDKs handle all three shapes; a hand-written client should check for `payload`.

Status	When
`400 Bad Request`	request validation failed (malformed body, bad parameter, failed business validation)
`404 Not Found`	the requested resource (hash, address, block) does not exist

1.4 Pagination

List endpoints accept these query parameters:

Param	Type	Range / default	Meaning
`page`	`Int`	`≥ 0`, default `0`	zero-based page index
`size`	`Int`	`1–100`, default `10`	page size
`sortBy`	`String`	field name (resource-dependent)	sort field
`sortDirection`	`String`	`ASC` / `DESC`, default `DESC`	sort order

The paged payload has this shape:

```
{
  "total": 42,
  "list": [ ],
  "page": 0,
  "pageSize": 10,
  "totalPages": 5
}
```

Field	Type	Meaning
`total`	`Long`	total matching records across all pages
`list`	`T[]`	records on this page
`page`	`Int`	echoed zero-based page index
`pageSize`	`Int`	echoed page size
`totalPages`	`Int`	total number of pages

1.5 Amounts

All monetary values (balances, fees, amounts, rewards, supply) are decimal (`BigDecimal`) with **8-decimal** scale. See the Yellow Paper for the `toScaledLong` scaling used in signed preimages. Native-token amounts are the exception: they are **integer base units** carried as decimal strings (see §7).

1.6 CORS and content type

- Requests and responses are JSON (`Content-Type: application/json`).
- CORS is permitted, so browser-based wallets and explorers may call the API cross-origin.

1.7 Non-custodial model

The node verifies signatures and applies state; it does **not** manage keys. There are **no** endpoints for key generation, seed phrases, mnemonics, or signing. Clients derive keys, build the preimage, hash, and sign entirely client-side, then submit finished transactions to the `POST` endpoints in §3–§9. See §12 and the Developer Guide / Yellow Paper for the signing pipeline.

---

2. Resource map

Resource group	Base path	Section
Wallets	`/api/v1/wallets`	§3
Transfer transactions	`/api/v1/transactions/transfers`	§4
Validator (registration) transactions	`/api/v1/transactions/validators`	§5
Exit-validator transactions	`/api/v1/transactions/exit-validators`	§6
Native tokens (transactions + state)	`/api/v1/transactions/tokens`, `/api/v1/tokens`	§7
Reward transactions (read-only)	`/api/v1/transactions/rewards`	§8
Main blocks	`/api/v1/blocks/main`	§9
Genesis blocks	`/api/v1/blocks/genesis`	§9
Validators (state)	`/api/v1/validators`	§10
Explorer	`/api/v1/explorer`	§11
Node info	`/api/v1/info`	§11
Transaction request bodies	(POST)	§12
Actuator	`/actuator`	§13
API docs (Swagger / OpenAPI)	`/docs`	§14

---

3. Wallets — `/api/v1/wallets`

Account-level read endpoints plus one validation helper. None of these expose or accept keys.

Method	Path	Params	Returns
GET	`/api/v1/wallets/{address}`	path `address`	confirmed balance (`BigDecimal`): confirmed − unconfirmed, floored at `0`
GET	`/api/v1/wallets/{address}/nonce`	path `address`	next expected nonce (`Long`) for the address
POST	`/api/v1/wallets/validateAddress`	body: address string	address validity / parse result

Path parameter:

Param	Type	Notes
`address`	`String`	`0x` + 40 lowercase hex chars

Example — balance:

```
curl http://localhost:7019/api/v1/wallets/0x06e1c0fa9955a700876f8cb0acc7f13fba9fb8ba
```

```
{ "timestamp": 1719700000000, "version": "1.1.0", "payload": "125.40000000" }
```

Example — nonce:

```
curl http://localhost:7019/api/v1/wallets/0x06e1c0fa9955a700876f8cb0acc7f13fba9fb8ba/nonce
```

```
{ "timestamp": 1719700000000, "version": "1.1.0", "payload": 0 }
```

The nonce returned is the **next** expected nonce for the account. The mempool allows **consecutive** nonces (pipelining): the next usable nonce = `confirmedNonce + pendingCount`.

---

4. Transfer transactions — `/api/v1/transactions/transfers`

Move coins, optionally with a memo (`data`, ≤ 256 bytes).

Method	Path	Query / body	Returns
GET	`/api/v1/transactions/transfers`	`address`, `unconfirmed` + pagination	paged transfers
GET	`/api/v1/transactions/transfers/{hash}`	path `hash`	one transfer by tx hash
POST	`/api/v1/transactions/transfers`	transfer tx body (§12)	accepts a transfer → `201 Created`

Query parameters (list):

Param	Type	Meaning
`address`	`String?`	filter by sender/recipient address
`unconfirmed`	`Boolean?`	include unconfirmed (mempool) transfers
`page`, `size`, `sortBy`, `sortDirection`	—	pagination (§1.4)

POST-specific body fields (in addition to the common fields in §12):

Field	Type	Notes
`amount`	`BigDecimal`	≥ `0.1` (`MINIMUM_AMOUNT`), unless a memo is attached
`recipientAddress`	`String?`	destination address
`data`	`String?`	optional memo, ≤ `256` bytes (`MAX_DATA_SIZE`)

Fee: ≥ `0.01` (`MINIMUM_FEE`).

---

5. Validator (registration) transactions — `/api/v1/transactions/validators`

Register the node as a validator. Admission is permissionless — run a node and submit a `PromoterTx` carrying the non-refundable deposit; there is no vote, approval, or lock-up. Note: the internal transaction/state TYPE names, the `*-promoter-tx` / `promoters-count` config keys, and the DB tables keep the legacy name **promoter** — the endpoints and JSON fields are validator.

Method	Path	Query / body	Returns
GET	`/api/v1/transactions/validators`	pagination	paged validator-registration txs
GET	`/api/v1/transactions/validators/{hash}`	path `hash`	one validator tx
POST	`/api/v1/transactions/validators`	validator tx body (§12)	registers → `201 Created`

POST-specific body fields:

Field	Type	Notes
`amount`	`BigDecimal`	registration deposit, exactly `2000` → treasury bank, **non-refundable**
`validatorKey`	`String`	must be unique

Fee: exactly `3`. The `2000` deposit is the primary Sybil guard; it goes to the treasury bank and is never returned (exiting does not refund it).

---

6. Exit-validator transactions — `/api/v1/transactions/exit-validators`

Gracefully retire a validator. An `ExitPromoterTx` removes the validator from the active set and slot schedule at the next epoch boundary. The original registration deposit stays non-refundable.

Method	Path	Query / body	Returns
GET	`/api/v1/transactions/exit-validators`	pagination	paged exit-validator txs
GET	`/api/v1/transactions/exit-validators/{hash}`	path `hash`	one exit-validator tx
POST	`/api/v1/transactions/exit-validators`	exit-validator tx body (§12)	retires → `201 Created`

POST-specific body fields:

Field	Type	Notes
`validatorKey`	`String`	the validator key being retired

There is no `amount` (the deposit is non-refundable). Fee: ≥ `0.01` (`MINIMUM_FEE`).

---

7. Native tokens — `/api/v1/transactions/tokens` and `/api/v1/tokens`

Native tokens (JZT-1) are optional user-defined fungible tokens that live alongside the native coin on the same chain, on a **separate ledger** — a token's supply is not part of the coin's fixed `100,000,000` and never touches emission, the treasury, or block rewards. Every token operation still pays its fee **in the native coin**. A single transaction type, `TokenTx`, carries an `op` byte selecting the operation; a companion set of read endpoints (§7.2) exposes token definitions, supply, and balances.

Authority: only a **registered validator** (a validator, active or standby) may `CREATE` a token; only the token's **issuer** (its creator) may `MINT`, `BURN`, or `SETCAP`; **anyone** holding a balance may `TRANSFER`. The new token's **`tokenId` is the CREATE transaction's hash** (globally unique; there is no symbol registry, so symbols are not unique).

7.1 Token transactions — `/api/v1/transactions/tokens`

Method	Path	Query / body	Returns
GET	`/api/v1/transactions/tokens`	`tokenId`, `unconfirmed` + pagination	paged token txs
GET	`/api/v1/transactions/tokens/{hash}`	path `hash`	one token tx by hash
POST	`/api/v1/transactions/tokens`	token tx body (§12)	accepts a token op → `201 Created`

Query parameters (list):

Param	Type	Meaning
`tokenId`	`String?`	filter by token id (the CREATE tx hash)
`unconfirmed`	`Boolean?`	include unconfirmed (mempool) token txs
`page`, `size`, `sortBy`, `sortDirection`	—	pagination (§1.4)

POST-specific body fields (in addition to the common fields in §12):

Field	Type	Notes
`op`	`Int`	operation: `0` CREATE, `1` MINT, `2` BURN, `3` SETCAP, `4` TRANSFER
`tokenId`	`String?`	target token (its CREATE tx hash); omit for CREATE
`symbol`	`String?`	CREATE: ticker, `1–12` chars
`name`	`String?`	CREATE: token name, `1–40` chars
`decimals`	`Int?`	CREATE: display decimals, `0–18` (display-only)
`cap`	`String?`	max supply, integer base-unit decimal **string** (CREATE / SETCAP)
`amount`	`String?`	integer base-unit decimal **string** (CREATE initial supply / MINT / BURN / TRANSFER)
`recipient`	`String?`	TRANSFER destination address

Token `cap` and `amount` are **integer base units** (like ERC-20 raw units), carried as their decimal **string** — send them as JSON strings to avoid precision loss; they are **not** the coin's 8-decimal scaled amounts, and never touch the coin supply. A token's own `decimals` is display-only metadata. Invariant enforced on both the live and sync paths: `0 ≤ totalSupply ≤ cap`, and `SETCAP` cannot drop the cap below the already-minted supply.

Fee (paid in the native coin): `CREATE` pays exactly `5` (`fee-create-token-tx`); `MINT`, `BURN`, `SETCAP`, and `TRANSFER` pay ≥ `0.01` (`MINIMUM_FEE`).

7.2 Token state — `/api/v1/tokens`

Read token definitions, supply, holders, and per-address balances. These are read-only; a token's supply and metadata change only through the `TokenTx` operations in §7.1.

Method	Path	Params	Returns
GET	`/api/v1/tokens`	pagination	paged token definitions (all tokens)
GET	`/api/v1/tokens/{tokenId}`	path `tokenId`	one token definition + current supply
GET	`/api/v1/tokens/{tokenId}/holders`	path `tokenId` + pagination	paged holder balances for the token
GET	`/api/v1/tokens/balances/{address}`	path `address` + pagination	paged token balances held by an address

A **token** object:

Field	Type	Meaning
`tokenId`	`String`	the CREATE tx hash (unique token id)
`symbol`	`String`	ticker (`1–12` chars; not unique)
`name`	`String`	token name (`1–40` chars)
`decimals`	`Int`	display decimals (`0–18`)
`cap`	`String`	maximum supply, integer base units
`totalSupply`	`String`	current minted supply, integer base units
`issuer`	`String`	creator address (the sole MINT/BURN/SETCAP authority)

A **balance** object:

Field	Type	Meaning
`tokenId`	`String`	the token id
`holder`	`String`	holder address
`balance`	`String`	held amount, integer base units

---

8. Reward transactions — `/api/v1/transactions/rewards` (read-only)

System-generated block rewards (bank → producing validator, 100% to the producer's wallet). These are **not** user-submittable; there is no `POST`.

Method	Path	Query / body	Returns
GET	`/api/v1/transactions/rewards`	pagination	paged reward txs
GET	`/api/v1/transactions/rewards/{hash}`	path `hash`	one reward tx

RewardTx carries fee `0`, nonce `0`, and a system payload; its value is validated against the canonical scheduled reward on every node.

---

9. Blocks — `/api/v1/blocks/main` and `/api/v1/blocks/genesis`

Two block resources share an identical shape. `main` is the user-transaction blocks; `genesis` is the epoch-boundary blocks that freeze the active validator set.

9.1 Main blocks — `/api/v1/blocks/main`

Method	Path	Params	Returns
GET	`/api/v1/blocks/main/{hash}`	path `hash`	a main block by hash
GET	`/api/v1/blocks/main/{hash}/previous`	path `hash`	the previous block
GET	`/api/v1/blocks/main/{hash}/next`	path `hash`	the next block
GET	`/api/v1/blocks/main`	pagination	paged main blocks

9.2 Genesis blocks — `/api/v1/blocks/genesis`

Method	Path	Params	Returns
GET	`/api/v1/blocks/genesis/{hash}`	path `hash`	a genesis block by hash
GET	`/api/v1/blocks/genesis/{hash}/previous`	path `hash`	the previous genesis block
GET	`/api/v1/blocks/genesis/{hash}/next`	path `hash`	the next genesis block
GET	`/api/v1/blocks/genesis`	pagination	paged genesis blocks

Common block fields include `timestamp`, `height`, `previousHash`, `hash`, `signature`, `publicKey`, plus per-type payload (main blocks carry the tx/state/receipt Merkle roots and `epochIndex`; genesis blocks carry the `epochIndex`, `activeAnchorCount`, and the frozen active-validator keys). See the Yellow Paper for exact layouts.

---

10. Validators (state) — `/api/v1/validators`

Read the validator set and election state. (Not to be confused with the validator *registration transaction* resource in §5.)

Method	Path	Params	Returns
GET	`/api/v1/validators`	pagination (by registration order)	all validators, ordered by registration (`createDate`, then id)
GET	`/api/v1/validators/active`	pagination	the current active set
GET	`/api/v1/validators/view`	pagination	validator view records

Selection is by **registration order**, not votes or rating — validators are ranked by `createDate` (longest-registered first). The active set is the still-privileged anchor validators plus the longest-registered elected validators, filled up to `promoters-count = 21`, minus offline/banned/exited nodes.

---

11. Explorer and node info

11.1 Explorer — `/api/v1/explorer/info`

Method	Path	Returns
GET	`/api/v1/explorer/info`	network/chain summary object

Payload fields:

Field	Type	Meaning
`nodesCount`	`Int`	number of known nodes
`blocksCount`	`Long`	total blocks
`transactionsCount`	`Long`	total transactions
`blockProductionTime`	`Number`	average block production time
`transactionsPerSecond`	`Number`	throughput (TPS)
`epochNumber`	`Long`	current epoch index
`epochDate`	`Long`	current epoch start time
`validatorsCount`	`Int`	number of validators
`maxSupply`	`BigDecimal`	fixed maximum supply (`100,000,000`)
`circulatingSupply`	`BigDecimal`	`maxSupply − bankBalance`

Example:

```
curl http://localhost:7019/api/v1/explorer/info
```


```
{
  "timestamp": 1719700000000,
  "version": "1.1.0",
  "payload": {
    "nodesCount": 3,
    "blocksCount": 1280,
    "transactionsCount": 4096,
    "blockProductionTime": 15000,
    "transactionsPerSecond": 0.0137,
    "epochNumber": 426,
    "epochDate": 1719600000000,
    "validatorsCount": 5,
    "maxSupply": "100000000.00000000",
    "circulatingSupply": "50000000.00000000"
  }
}
```

11.2 Node info — `/api/v1/info`

Method	Path	Returns
GET	`/api/v1/info`	`{ nodeKey, host, port, networkId, genesisHash, chainSpecDigest, version, apiVersion, protocolVersion, syncStatus, faucet }`
GET	`/api/v1/info/version`	`{ version, apiVersion, protocolVersion }` — same meanings as `/info`: build, REST/RPC API, wire protocol (until node 0.1.10 `protocolVersion` here carried the API version, F-057)
GET	`/api/v1/info/uptime`	node uptime (ms)
GET	`/api/v1/info/genesis`	the chain's public `genesis.json` — anchors, final genesis balances, signer public key, pinned bootstrap hashes + signatures; a downloaded node boots from it (`--janzeer.genesis.file`)

`/api/v1/info` payload fields:

Field	Type	Meaning
`nodeKey`	`String`	this node's public key (its P2P identity / UID; the **validator key** you register)
`host`	`String`	advertised external host
`port`	`Int`	advertised external P2P port
`networkId`	`String`	`consensus.network-id` — `janzeer` on the mainnet (a private test network uses `janzeer-testnet`); sign every transaction with it
`genesisHash`	`String`	hash of the epoch-1 genesis block: the chain's identity
`chainSpecDigest`	`String`	consensus rule-set + genesis digest exchanged in the P2P handshake
`version` / `apiVersion` / `protocolVersion`	`String`	node software / REST-RPC API / P2P wire protocol versions
`syncStatus`	`String`	`SYNCHRONIZED`, `PROCESSING` or `NOT_SYNCHRONIZED`
`faucet`	`Boolean`	this node runs a faucet (`POST /api/v1/faucet`, §11.3) — only possible on a private test network; always `false` on the mainnet

11.3 Faucet — `/api/v1/faucet` (private test networks only)

Present only on a private test network you run yourself — there is no public testnet and no public faucet (a node refuses to run a faucet on the mainnet id; `GET /api/v1/info` → `faucet` says whether it is there).

Method	Path	Returns
GET	`/api/v1/faucet`	`{ enabled, address, amount, cooldownHours }`
POST	`/api/v1/faucet` body `{ address }`	`201` `{ hash, address, amount }` — the faucet signs and submits an ordinary transfer; `400` invalid address; `404` no faucet on this node; `429` + `Retry-After` (`type: FAUCET_COOLDOWN`) when the address **or** the client IP already received a drip inside the cooldown window

---

12. Transaction request bodies (POST)

All transaction `POST` endpoints (§4–§7) accept the same **common fields**, plus a small set of **type-specific fields**. The client builds and signs the transaction; the node only verifies it (non-custodial — there are no key or signing endpoints). For the exact preimage byte layout and signing algorithm, see the Yellow Paper and the Developer Guide.

12.1 Common fields (every tx type)

Field	Type	Meaning
`timestamp`	`Long`	client build time (epoch milliseconds), part of the signed preimage
`fee`	`BigDecimal`	transaction fee (per-type minimum/exact value — see below)
`nonce`	`Long`	sender's account nonce (replay protection)
`hash`	`String`	`doubleSHA256(preimage)`, lowercase hex
`senderAddress`	`String`	`0x` + 40 lowercase hex chars
`senderSignature`	`String`	ECDSA secp256k1, RFC-6979, low-S, DER then Base64, over the raw 32-byte digest
`senderPublicKey`	`String`	33-byte compressed public key, hex

12.2 Type-specific fields and fee/amount rules

Tx type	Endpoint	Extra fields	Fee	Amount / deposit
Transfer	`POST /api/v1/transactions/transfers`	`amount`, `recipientAddress?`, `data?`	≥ `0.01`	`amount` ≥ `0.1` unless a memo is attached
Promoter	`POST /api/v1/transactions/validators`	`amount`, `validatorKey`	`3`	deposit exactly `2000` → bank (non-refundable)
ExitPromoter	`POST /api/v1/transactions/exit-validators`	`validatorKey`	≥ `0.01`	—
Token	`POST /api/v1/transactions/tokens`	`op`, `tokenId?`, `symbol?`, `name?`, `decimals?`, `cap?`, `amount?`, `recipient?`	CREATE exactly `5`, others ≥ `0.01`	token `cap` / `amount` are integer base-unit decimal **strings** (not the coin), paid fee in the coin

Constants: `MINIMUM_FEE = 0.01`, `MINIMUM_AMOUNT = 0.1`, `MAX_DATA_SIZE = 256` bytes, `fee-create-token-tx = 5`. Native-token `cap` and `amount` are integer base units sent as JSON **strings** (§7). RewardTx (§8) is system-generated and never POSTed.

All successful tx POSTs return `201 Created` with the accepted transaction in `payload`.

12.3 Example — submit a transfer

```
curl -X POST http://localhost:7019/api/v1/transactions/transfers \
  -H 'Content-Type: application/json' \
  -d '{
    "timestamp": 1719700000000,
    "fee": "0.01",
    "nonce": 0,
    "hash": "a1b2c3...",
    "senderAddress": "0x06e1c0fa9955a700876f8cb0acc7f13fba9fb8ba",
    "senderSignature": "MEUCIQD...base64...",
    "senderPublicKey": "02a1b2c3...compressed-hex...",
    "amount": "1.5",
    "recipientAddress": "0x1111111111111111111111111111111111111111",
    "data": "",
    "nonce": 4
  }'
```

```
{
  "timestamp": 1719700000001,
  "version": "1.1.0",
  "payload": {
    "hash": "a1b2c3...",
    "senderAddress": "0x06e1c0fa9955a700876f8cb0acc7f13fba9fb8ba",
    "recipientAddress": "0x1111111111111111111111111111111111111111",
    "amount": "1.5",
    "fee": "0.01",
    "nonce": 0
  }
}
```

A validation failure (e.g. bad signature, insufficient balance, fee below minimum) returns `400` with the error shape in §1.3.

---

13. Actuator — `/actuator` (unwrapped)

Spring Boot Actuator endpoints. These are **not** wrapped in the response envelope (§1.2).

Method	Path	Purpose
GET	`/actuator/health`	liveness / health status
GET	`/actuator/info`	build/app info
GET	`/actuator/metrics`	metrics index and individual metrics
GET	`/actuator/prometheus`	Prometheus scrape endpoint

---

14. API docs — `/docs` (unwrapped)

Interactive and machine-readable API documentation. These paths are **not** enveloped.

Method	Path	Purpose
GET	`/docs/swagger-ui.html`	interactive Swagger UI
GET	`/docs/api-docs`	OpenAPI document (JSON) served live by the node

A machine-readable **OpenAPI spec is served live at** `/docs/api-docs`. A static copy may also be included in the repository at `docs/api/openapi.json` for offline tooling and SDK generation; when both are present, the live `/docs/api-docs` document is authoritative for the running node version.

---

15. Quick reference — all endpoints

Method	Path	Resource
GET	`/api/v1/wallets/{address}`	balance
GET	`/api/v1/wallets/{address}/nonce`	next nonce
POST	`/api/v1/wallets/validateAddress`	validate address
GET	`/api/v1/transactions/transfers`	list transfers
GET	`/api/v1/transactions/transfers/{hash}`	one transfer
POST	`/api/v1/transactions/transfers`	submit transfer
GET	`/api/v1/transactions/validators`	list validator txs
GET	`/api/v1/transactions/validators/{hash}`	one validator tx
POST	`/api/v1/transactions/validators`	register validator
GET	`/api/v1/transactions/exit-validators`	list exit-validator txs
GET	`/api/v1/transactions/exit-validators/{hash}`	one exit-validator tx
POST	`/api/v1/transactions/exit-validators`	retire validator
GET	`/api/v1/transactions/tokens`	list token txs
GET	`/api/v1/transactions/tokens/{hash}`	one token tx
POST	`/api/v1/transactions/tokens`	submit token op
GET	`/api/v1/tokens`	list token definitions
GET	`/api/v1/tokens/{tokenId}`	one token (definition + supply)
GET	`/api/v1/tokens/{tokenId}/holders`	token holders
GET	`/api/v1/tokens/balances/{address}`	an address's token balances
GET	`/api/v1/transactions/rewards`	list reward txs
GET	`/api/v1/transactions/rewards/{hash}`	one reward tx
GET	`/api/v1/blocks/main`	list main blocks
GET	`/api/v1/blocks/main/{hash}`	one main block
GET	`/api/v1/blocks/main/{hash}/previous`	previous main block
GET	`/api/v1/blocks/main/{hash}/next`	next main block
GET	`/api/v1/blocks/genesis`	list genesis blocks
GET	`/api/v1/blocks/genesis/{hash}`	one genesis block
GET	`/api/v1/blocks/genesis/{hash}/previous`	previous genesis block
GET	`/api/v1/blocks/genesis/{hash}/next`	next genesis block

Method	Path	Resource
GET	`/api/v1/validators`	validators by registration order
GET	`/api/v1/validators/active`	active set
GET	`/api/v1/validators/view`	validator view records
GET	`/api/v1/explorer/info`	network summary
GET	`/api/v1/info`	node identity
GET	`/api/v1/info/version`	node, API and wire-protocol versions
GET	`/api/v1/info/uptime`	node uptime
GET	`/actuator/health`	health (unwrapped)
GET	`/actuator/info`	build info (unwrapped)
GET	`/actuator/metrics`	metrics (unwrapped)
GET	`/actuator/prometheus`	Prometheus (unwrapped)
GET	`/docs/swagger-ui.html`	Swagger UI (unwrapped)
GET	`/docs/api-docs`	OpenAPI document (unwrapped)

Document status: version 1.1. Reflects node version 0.1.0. Exact byte-level facts (preimage layouts, hashing, amount scaling, derivation constants) live in the Yellow Paper.

15. JSON-RPC 2.0 — `POST /rpc` and `ws://…/rpc/ws`

The same services and intake rules as the REST endpoints above, exposed as JSON-RPC 2.0 for exchange / custody integrations. Standard envelope and error codes, params by name or by position, batches, and — over the WebSocket — `janzeer_subscribe` for `newBlocks` (with replay from a height) and `addressActivity` (every final transaction touching a watched address). The REST envelope of §1.2 is not applied; `GET /rpc` lists the methods. Method table, error codes and a worked example: Developer Guide §2.14.

Methods: `janzeer_methods`, `janzeer_getInfo`, `janzeer_getChainSpec`, `janzeer_getStats`, `janzeer_getAccount`, `janzeer_getBalance`, `janzeer_getNonce`, `janzeer_getTokenBalances`, `janzeer_getTransactionsByAddress`, `janzeer_sendTransfer`, `janzeer_sendToken`, `janzeer_sendRegisterValidator`, `janzeer_sendExitValidator`, `janzeer_sendRawTransaction`, `janzeer_getTransactionByHash`, `janzeer_waitForFinality`, `janzeer_estimateFee`, `janzeer_getTip`, `janzeer_getBlockByNumber`, `janzeer_getBlockByHash`, `janzeer_getBlocks`, `janzeer_getToken`, `janzeer_listTokens`, `janzeer_getTokenHolders`, `janzeer_listValidators`, `janzeer_getActiveValidators`, `janzeer_getValidator`, `janzeer_getEpoch`; WebSocket only: `janzeer_subscribe`, `janzeer_unsubscribe`.