JANZEER BLOCKCHAIN

White Paper

Vision, economics & consensus

Document v1.0 · Janzeer Network · 2026

Janzeer White Paper

Janzeer is a fixed-supply, Byzantine-fault-tolerant, permissionless Layer-1 blockchain with a non-custodial design at its core: the node verifies signatures and never holds, generates, or stores user keys. Its native coin has a hard cap of 100,000,000 units that are never minted — every coin exists at genesis, and block rewards are paid out of a pre-funded on-chain treasury rather than created from nothing. There is no staking, delegation, or voting: block production is carried out by validators who register permissionlessly behind a non-refundable deposit, produce blocks in a fixed rotation, and finalize each block through a two-phase BFT vote. This paper explains what Janzeer is, the problem it addresses, its economic model, how transactions and fees work, and how consensus, validator admission, and security fit together — using the real parameters from the codebase and stating its current development caveats honestly.

1. What Janzeer Is and the Problem It Addresses

Most blockchains force users into one of two uncomfortable positions. Either keys are held by a custodian — an exchange or a hosted wallet — which reintroduces exactly the counterparty risk that public blockchains were meant to remove, or the network's economics rely on perpetual inflation, quietly diluting holders to pay for security. Janzeer is built to avoid both.

Janzeer is a Layer-1 network that combines:

- A **non-custodial** architecture in which private keys are derived and used only on the client side. The node has no seed-generation or key-generation endpoints; it only verifies signatures and applies state transitions.
- A **fixed, never-minted supply** of 100,000,000 coins, with security rewards funded by an on-chain treasury and recycled transaction fees rather than by inflation.
- **Permissionless, stake-free** block production combined with **two-phase BFT finality**, so that blocks are final the moment they are committed — there is no probabilistic confirmation depth to wait out.

The result is a network where holders keep custody of their own assets, the total supply is predictable and capped, and finality is fast and deterministic. Anyone who runs a node and posts a one-time non-refundable deposit can become a validator — there is no vote, approval, or lock-up to negotiate. The technology stack is Kotlin on Spring Boot and the JVM; the node software is version 0.1.0. The native coin has no ticker symbol defined in code — this paper refers to it as "the native coin."

2. Non-Custodial Design Philosophy

Janzeer treats key management as a client responsibility and verification as the node's only job. This is a deliberate, enforced boundary, not a convention.

- **Keys never reach the node.** Wallets are hierarchical-deterministic (HD): a recovery phrase is expanded into a seed and then into a key tree entirely on the user's device. The node exposes no endpoint that generates seeds or keys, and it never stores a private key.
- **The client builds, signs, and submits; the node verifies.** A transaction is constructed and signed on the client. The node receives a fully formed, signed transaction, checks the signature against the sender's public key and address, checks the nonce and balance, and either accepts or rejects it. Signatures use ECDSA over the secp256k1 curve with deterministic RFC-6979 nonces and enforced canonical low-S form.
- **Addresses are derived deterministically and verifiably.** A Janzeer address is `"0x"` followed by the first 20 bytes of the Keccak-256 hash of the 33-byte compressed public key. The canonical stored form is all-lowercase hex; a real **EIP-55** mixed-case checksum is applied at the wallet/API display boundary and validated on input, but it is presentational and not consensus-relevant. Anyone can recompute an address from a public key, and the cross-platform parity test vectors committed in the repository prove byte-for-byte agreement between the node and independent wallet implementations (HD derivation, address, transaction hash, and signature all match).

Because the node holds no keys, there is no honeypot of user secrets to steal, and a compromised node cannot move user funds. This boundary is the foundation of Janzeer's security model.

Every signed transaction also binds a **network-id** (default `janzeer`, distinct per chain) into its preimage, so a transaction signed for one network can never be replayed on another.

3. Economic Model: Fixed Supply and the C + D Emission

Janzeer's economics are designed around a single principle: **zero inflation**. The supply is fixed, coins are never minted, and validators are still perpetually rewarded. This is achieved by combining a declining treasury-funded subsidy (C) with treasury fee recycling (D).

Fixed supply and genesis allocation

The maximum supply is `100,000,000` coins (held internally at 8-decimal scale) and is fully allocated at genesis. There is no minting function anywhere in the protocol. The genesis allocation sums to **exactly** the maximum supply, enforced by a fail-fast cap-invariant assertion at startup:

Allocation	Amount	Notes
Treasury "bank" `0x000…000`	77,500,000	Key-less, capture-proof
Ecosystem / distribution pool	12,000,000	Faucet / airdrop / governed
Founder	7,500,000	Carved from treasury when a custody address is supplied
Anchor validators (3 × 1,000,000)	3,000,000	Bootstrap validators (see §5)

Because the treasury address has no private key, no one can spend it directly; only the protocol's own reward and fee logic can move coins out of or into it. The founder and distribution allocations (and the optional dev-fund faucet) are carved out of the treasury only when a custody address is supplied; absent one, they fold back into the treasury, so nothing is ever minted beyond the cap. The circulating supply is defined as `MAX_SUPPLY − bankBalance`.

C — declining block subsidy

Each block carries a scheduled subsidy that halves over time and never falls below a perpetual floor:

```
scheduledReward(h) = max(rewardTailFloor, rewardBlock / 2^(h / rewardHalvingPeriod))
```

With `rewardBlock = 10`, a halving period of `2,100,000` blocks, and a tail floor of `0.5`, the subsidy starts at 10 coins per block and halves every 2,100,000 blocks. Because blocks are produced every 15 seconds, that halving period is **roughly one year**, so the block reward effectively halves yearly before settling at a perpetual minimum of 0.5 coins per block. The amount actually paid at a given height is `min(scheduledReward(h), bankBalance)`: when the treasury empties, scheduled pay drops to zero unless the treasury is refilled.

D — treasury fee recycling

The treasury is refilled by transaction fees. A fixed share of every fee returns to the bank (`treasuryFeeShareBps = 2000`, i.e. **20% of every fee → treasury**), while the remaining **80% goes to the block producer's wallet**. In addition, the non-refundable deposit paid to register a validator flows to the treasury. This recycling means the treasury is continually topped up by network activity, so the perpetual tail subsidy can be sustained indefinitely without ever minting a new coin. The net effect of C + D is a security budget that declines predictably, never inflates the supply, and remains perpetual.

The block reward is paid **100% to the producing validator's wallet** — there is no delegation and no commission split. The reward is delivered by a system-generated transaction (fee 0, nonce 0) whose value every node independently recomputes against the canonical schedule; a block whose reward does not match the canonical value is rejected.

4. Transactions and Fees

Janzeer uses an account model. Each account has a spendable `balance` and a `nonce` for replay protection. Amounts are tracked at 8-decimal precision. The protocol-wide minimums are a `MINIMUM_AMOUNT` of `0.1`, a `MINIMUM_FEE` of `0.01`, and a `MAX_DATA_SIZE` of `256` bytes for an attached transfer memo.

Replay and overspend are prevented at the protocol level. Every transaction carries a per-account nonce that is included in both the signed bytes and the account's state hash; the mempool permits consecutive nonces for pipelining, and apply-time enforcement is strict. An account balance can never go negative, on both the live path and during sync, and the mempool subtracts already-pending spends before accepting a new transaction.

The transaction types and their fees:

Type	Purpose	Fee	Amount	Notes
TransferTx	Move coins, with an optional memo (≤256 bytes)	≥ 0.01	—	Amount ≥ 0.1 unless a memo is attached
PromoterTx	Register a node as a validator	3 (exact)	2000 (→ treasury, non-refundable)	Validator key must be unique
ExitPromoterTx	Gracefully retire a validator	≥ 0.01	—	Removed from the active set next epoch; deposit stays gone
RewardTx	Pay a block reward (treasury → producer)	0 (system)	—	Not user-submitted
SlashingTx	Prove equivocation by a validator	Fee-free (system)	—	Bans offender, pays reporter a bounty

Fees are split, never burned. Of every fee, 20% returns to the treasury and 80% goes to the block producer. This keeps the supply intact while rewarding producers directly and recycling value back into the long-term security budget. The `2000`-coin PromoterTx deposit is non-refundable and is the network's primary Sybil guard — retiring a validator does not return it.

Janzeer also supports optional **user-defined fungible tokens** (JZT-1) alongside the native coin. Any registered validator can issue a token with a capped, adjustable supply that it alone mints, burns, or re-caps, and anyone can hold and transfer it. Tokens live on a separate ledger carried by a single `TokenTx` type: they never mint or affect the native coin — its `100,000,000` cap is untouched — and each token operation simply pays its fee in the native coin. The exact token transaction layout lives in the Yellow Paper.

5. Consensus and Validator Admission

Janzeer's consensus family is **permissionless validator block production with two-phase (PREPARE → COMMIT) BFT finality**, in the style of PBFT/Tendermint. Producers are called **validators**.

Permissionless, stake-free admission

There is no staking, delegation, or voting anywhere in Janzeer. Becoming a validator is permissionless: **run a node and submit a PromoterTx carrying the `2000` non-refundable deposit** (paid to the treasury). There is no vote, no approval, and no lock-up. The deposit, together with a one-seat-per-key rule, is the Sybil guard. Block rewards go 100% to the producer — they pay for the *work* of running the chain, not for yield on capital.

A validator leaves gracefully by submitting an **ExitPromoterTx**, which removes it from the active set and slot schedule at the next epoch boundary (so it stops being assigned slots immediately, rather than degrading liveness until it is ejected for downtime). The registration deposit remains non-refundable on exit.

Selection by registration order

Eligible validators are ranked by **registration order** — their `createDate` (then id), longest-registered first — not by any rating, stake, or vote weight. While the number of registered validators is at or below the seat cap, all of them are active; any overflow rotates fairly through a standby bench (below).

Slots, epochs, and the active set

Time is divided into fixed **slots**. Each full slot is 15,000 ms (15 seconds): an active production window of `time-slot-duration = 12,000 ms` followed by an intermission of `time-slot-interval = 3,000 ms`. The slot owner is chosen by deterministic round-robin over the epoch's frozen list of active validators.

An **epoch** is `epoch-height = 17,000` main blocks (**≈ 3 days** at 15-second blocks). At each epoch boundary a GenesisBlock is produced that re-elects the active set and increments the epoch index. The **active set** is composed of:

- Still-**privileged anchor** validators (see below), plus
- The longest-registered elected validators, filling the set up to `promoters-count = 21`, minus any offline, banned, or exited members.

Within that set, `rotating-producer-seats = 5` elected seats rotate round-robin through a standby bench of up to `producer-candidate-pool-size = 100` candidates in registration order, giving fair turnover. Validators that stay silent on their assigned slot are treated as downtime and ejected the following epoch.

Automatic anchor phase-out

Janzeer bootstraps from three **anchor** validators, but decentralizes itself automatically. Anchors start privileged — always in the active set and immune to downtime ejection — and shed that privilege **one per epoch** (`anchor-demotions-per-epoch = 1`) once there are enough **proven independent** validators: non-anchor validators that produced in every one of the last `validator-proven-epochs = 2` epochs. Counting sustained, proven producers rather than raw registrations means a Sybil flood cannot weaponize the mechanism.

The healthy validator set is never allowed to drop below `safety-floor-validators = 4`. If the independent set later collapses below that floor, dormant anchors automatically **re-engage** — the system is anti-fragile. Demotion is not removal: a demoted anchor keeps validating on merit like any other registrant. The number of privileged anchors is committed in each GenesisBlock (`activeAnchorCount`) so the set is reconstructable on every node.

Two-phase BFT finality

Finality uses a quorum defined over the actual active-set size N:

```
bftQuorum(N) = floor(2N / 3) + 1
```

This is the familiar "2f+1" threshold when N = 3f+1. The protocol runs in two phases. Each active validator casts a signed **PREPARE** vote on a valid pending block; once PREPARE reaches quorum, each validator fully validates the block and casts a **COMMIT** vote; once COMMIT reaches quorum, the block is **final**. This is **single-block finality** — there is no confirmation-depth waiting period. Votes are bundled into a **QuorumCertificate** and gossiped once per slot. A minimum of **4** validators (N ≥ 3f+1, tolerating one fault) is required to launch.

Equivocation and slashing

A validator may cast only one vote per slot per stage, tracked by the node. A conflicting second vote is provable equivocation (a double-vote) and produces a SlashingTx. Reaching `equivocation-ban-threshold = 2` distinct proven offenses results in a **permanent ban**, and the reporter receives a bounty of `slash-reporter-bounty = 5` from the treasury. Slashing is therefore evidence-based: misbehavior is cryptographically proven, not merely alleged.

Fork choice, clocks, and rule-set integrity

Fork choice is longest-valid-chain with a single tip that only moves forward. A rollback that would discard already-committed state requires the same 2f+1 quorum of distinct authenticated validators to corroborate it; Janzeer is a single-tip system, not a general reorg engine. Nodes are NTP-synchronized (`ntp-offset-threshold = 250 ms`, tightened for the 15-second wall-clock slots) and a node halts production if its clock is not synchronized or if it is not fully synced. Each block can hold up to `block-capacity = 100` transactions;

within a block the reward transaction applies first, then user transactions in ascending nonce order, making application fully deterministic.

The consensus-critical parameters (everything under `consensus.*`) form a canonical rule-set fixed at genesis. Each node computes a **chain-spec digest** over those parameters and exchanges it during the P2P handshake; a peer whose digest differs is rejected. A node that edits any consensus parameter therefore forks itself off the network instead of silently diverging.

Consensus parameters at a glance

Parameter	Value	Meaning
`MAX_SUPPLY`	100,000,000	Hard cap, fully allocated at genesis, never minted
Treasury allocation	77,500,000	Pre-funded bank at `0x000…000`
`rewardBlock`	10	Initial scheduled block subsidy
`rewardHalvingPeriod`	2,100,000 blocks	Subsidy halving interval (≈ 1 year at 15-s blocks)
`rewardTailFloor`	0.5	Perpetual minimum subsidy
`treasuryFeeShareBps`	2000 (20%)	Fee share recycled to treasury (80% to producer)
`MINIMUM_AMOUNT`	0.1	Minimum transfer amount (without memo)
`MINIMUM_FEE`	0.01	Minimum transaction fee
`MAX_DATA_SIZE`	256 bytes	Maximum attached memo size
`fee-promoter-tx`	3	Fee to register a promoter
`amount-promoter-tx`	2000	Non-refundable registration deposit (→ treasury)
`promoters-count`	21	Active-set cap
`rotating-producer-seats`	5	Rotating elected seats
`producer-candidate-pool-size`	100	Standby candidate bench
`epoch-height`	17,000	Main blocks per epoch (≈ 3 days)
`time-slot-duration`	12,000 ms	Active production window
`time-slot-interval`	3,000 ms	Slot intermission
Full slot	15,000 ms (15 s)	Total slot length
`bftQuorum(N)`	floor(2N/3)+1	Finality quorum (2f+1)
`equivocation-ban-threshold`	2	Proven offenses → permanent ban
`slash-reporter-bounty`	5	Bounty paid to a slashing reporter
`safety-floor-validators`	4	Healthy set never demoted below this
`anchor-demotions-per-epoch`	1	Anchor privileges shed per epoch

Parameter	Value	Meaning
`validator-proven-epochs`	2	Epochs a validator must produce in to count as proven
`ntp-offset-threshold`	250 ms	Clock-sync tolerance
`block-capacity`	100	Maximum transactions per block

6. Security Properties

Janzeer's security rests on several mutually reinforcing properties:

- **Non-custodial by construction.** Keys exist only on the client. There is no seed or key-generation endpoint, so there is no central store of user secrets to compromise, and a compromised node cannot spend user funds.
- **Replay and overspend protection.** Every transaction's per-account nonce is bound into both the signed preimage and the account state hash, and the network-id is bound into the preimage to block cross-chain replay. Balances can never go negative, on either the live or sync path.
- **Deterministic, canonical signatures.** ECDSA over secp256k1 with RFC-6979 deterministic nonces and enforced low-S removes signature malleability and nonce-reuse risk.
- **Sybil resistance without stake or PoW.** Admission is gated by a non-refundable deposit and one seat per key, tempered by fair rotation and a conservative anchor phase-out that counts only proven, sustained validators and self-corrects by re-engaging anchors if the set collapses.
- **BFT safety.** The network tolerates up to f Byzantine validators out of N = 3f+1, with finality requiring a `floor(2N/3)+1` quorum. Equivocation is provable and slashable, with a permanent ban after two proven offenses.
- **Authenticated peer-to-peer.** Peers complete a mutual challenge-response handshake, verify each other's ECDSA signatures, and compare chain-spec digests before being admitted; only authenticated peers running the same rule-set are added.
- **Key-less treasury.** The genesis treasury has no private key, so it can never be spent directly — only the protocol's reward and fee logic can move its coins. Config-driven genesis keeps committed defaults reproducible across all nodes.

7. Honest Caveats

In keeping with an accurate, no-hype presentation, the following limitations and defaults should be understood plainly:

- **Dev/testnet anchor defaults.** The anchor validator secrets and genesis defaults committed in the repository are for development and private test networks. They must be rotated for any production deployment.

- **Dev-fund faucet is a testing aid.** An optional development pre-fund exists for end-to-end testing on private networks. It is off by default and must remain off in production.
- **Ecosystem pool custody must be transparent.** The 12,000,000-coin ecosystem/distribution pool should be held under transparent custody (faucet, airdrop, or governed process), not in a founder wallet.
- **Finality requires a live quorum.** A single node cannot finalize blocks. Finality depends on a live quorum of the active validator set, so a real network with a sufficient set of active, honest validators is required for the BFT guarantees to hold.
- **The registration deposit is non-refundable by design.** Exiting a validator, gracefully or otherwise, does not return the deposit.

These caveats are properties of the current configuration and deployment posture, not of the protocol's design intent, and they should be addressed by anyone standing up a production network.

Document status. Version 1.0. This white paper reflects Janzeer node version 0.1.0. Exact byte-level facts — cryptographic primitives, key and address derivation, transaction and block byte layouts, hashing, and amount scaling — live in the Yellow Paper.