MINTID

MintID Validator Manual

Operating consensus for an identity-first chain: economics, slashing, node operations and what the bootstrap deal actually is.

VERSION 0.1 · 2026-08-09 · PUBLIC EDITION

PRE-GENESIS — SET SIZE, BOND, UNBONDING AND SLASH FRACTIONS ARE OPEN PARAMETERS

ECONOMIC FIGURES ARE ILLUSTRATIVE BANDS, PENDING EXTERNAL SIMULATION

> **Status and scope.** Public edition (v0.1). There is no mainnet, no genesis, no token price and no public validator set; the only live network is a single-node internal testnet (MINT-212). Validator set size, minimum self-bond, voting-power cap, unbonding period, jailing thresholds and slash fractions are **open parameters that must be finalized before genesis** — figures quoted from the testnet are Cosmos SDK defaults, not ratified values. Economic figures are illustrative bands pending the external Phase-0 simulation.

1. What makes validating MintID different

MintID is a sovereign delegated-proof-of-stake L1 (Cosmos SDK v0.54.3 + CometBFT v0.39.3) whose product is **identity trust, not DeFi**. The commercial word "miner" survives in marketing; the technical role is a validator.

What you validate: deterministic protocol state transitions — issuer and verifier registries, 32-byte status roots arriving as a steady heartbeat, bonds, governance and token movements.

What you never touch: KYC documents, biometrics, names, addresses, presentation data. Validators validate cryptographic evidence only; no reward exists for subjective "identity validation work". The state you replicate contains, by construction, no PII, no credential payloads, no presentations and no per-user records (R2, §8.1).

Why the set matters more than usual: a consensus takeover would compromise issuer status roots and registry integrity — the trust fabric itself — not just token balances.

The determinism boundary is absolute: consensus code never makes network calls, never queries a KYC system, never reads the wall clock outside block time, never calls an HSM, RPC, sidecar or external database during block execution. Every validator must reach the same result from the transaction bytes and current state alone. Cross-platform consensus replay differences must be zero before mainnet.

2. Chain and consensus parameters

Genesis candidates (SPEC §5.1 — subject to simulation and testnet benchmarking):

PARAMETER	CANDIDATE VALUE
Active validator set	max 100, top-N by capped bonded voting power
Target block time	2–3 s
Finality SLO	p95 ≤ 5 s under normal conditions
Admission	permissionless: run node → bond → register → top-N
Voting power	capped per validator to limit concentration
Minimum self-bond / unbonding period / jailing thresholds	defined in genesis after economic simulation

Note: the famous **10 seconds** is the holder presentation lifetime (R14) — it is *not* a chain finality requirement.

On today's testnet (SDK defaults, placeholders): 21-day unbonding, `max_validators` 100, denom `stake`, ed25519 consensus keys, ~5 s blocks (`timeout_commit`), community tax 2%. Chain-id `mintid-testnet-1`, bech32 prefix `mintid` — all placeholders until gate G0-2. Address prefixes: `mintid1…`, `mintidvaloper1…`, `mintidvalcons1…`.

3. Economics: what a validator actually earns

Supply and emission

- **Hard cap 108,000,000 tokens, immutable** (R17). Burns never replenish the reward pool; the cap counts total-ever-minted.
- Genesis bands (ratified, ADR-0005): development 15%, public sale 20–25% (MiCA Title II), and **reward pool ≥ 60% of the cap (≥ 64.8 M tokens)** – the security budget is the dominant allocation, and it has **exactly one use**: validator emission. Genesis dev/foundation accounts neither delegate nor vote.
- Per-epoch emission declines on a published schedule and materializes pro-rata per block, rounding down. Code placeholders pending simulation (G0-3): initial epoch = 1% of cap, ×0.90 decay per yearly epoch → ~1,080,000 tokens in year 1.

The reward formula (SPEC §5.2)

```
W_i = P_i × S_i        # capped voting power × signed-vote performance (0..1)
R_i = E(epoch) × W_i / ΣW
```

Rewards flow to validators **and their delegators** through standard `x/distribution`; you charge your declared commission. Missed votes cost you the performance share and can jail you.

The bootstrap deal (ADR-0006 – read this before budgeting)

Pre-listing, emission has no market price, and MintID's policy forbids *promised* token rewards outside the schedule. The ratified answer: bootstrap validators are compensated through **service agreements denominated in fiat/stablecoin, paid from the operating budget (sale proceeds) for the first 12–24 months**. Key terms:

- Agreements must **not** give the operating group control over your keys or votes – key independence is what makes your stake count as independent for the launch gate.
- Contracted stake is publicly disclosed as such (genesis-validator registry with disclosed self-bonds and agreement status).
- Illustrative draft bands: service agreement **$30–60k/year** against professional infra costs of **$25–60k/year**; sector all-in benchmark $50–150k/operator-year. Year-1 emission share ≈ 10,800 tokens at the placeholder split. **Without a token price, the validator business in years 1–2 is the service agreement; the token is optionality.**

After emission ends

The long-term budget is the fee market: transaction fees plus the validator-directed share of the protocol cuts on agent mint/lease (flow A) and consensual disclosure (flow C). The published successor ladder if that falls short: adjust cut points within bands → activate protocol-state rent → supplementary shared security. **A post-cap mint is prohibited, permanently.** The consensus firewall also holds in the other direction: off-chain service revenue (verification APIs, arbiter fees) never influences consensus rewards (R26), and verification stays free (R15).

4. Slashing and jailing

EVENT	TREATMENT (SPEC §5.3)
Double-sign / equivocation	Objective evidence; **slash + permanent tombstone**
Sustained downtime	Objective missed-block evidence; **slash + temporary jail**
Invalid state transition	Rejected before finality; no reward
Consensus-key compromise	Emergency rotation + incident process; possible jail pending evidence
Alleged censorship	No MVP slashing rule (attribution too subjective)

- **There is no identity-specific validator slashing.** Issuer fraud runs through the issuer bond and council adjudication (`x/issuerbond`), never through your stake.
- Slash fractions are open pre-genesis parameters. The testnet carries SDK defaults only: 5% double-sign, 1% downtime, 100-block window at 50% minimum, 10-minute jail.
- A defined portion of validator slashes is burned (R17); the split is undefined.

5. Hardware and why state stays bounded

R19 is the sizing promise: live state grows as `O(issuers + verifier origins + validators + revocation roots + governance records)` — and does **not** grow with people, credentials or presentations. No state table contains a holder's name, account, serial, KYC record or proof history.

- **The real load driver is heartbeats:** every active issuer publishes a status root every ≤30 s — 2,880 tx/day per issuer, forever. Issuer count is small by design (economic scenarios use 2/6/15 issuers).
- Ordinary validators retain current state plus a defined short history; archival nodes keep full history (R27). Testnet pruning: `default` (last ~362k states); state-sync snapshots not yet enabled.
- **App DB must be** `pebbledb` — goleveldb breaks iavl's empty-root convention on never-written stores and panics the node on restart (MINT-64).
- No hardware-class table exists yet; node bootstrap times will be measured and published per hardware class before mainnet. Only observed data point: the whole testnet (validator + verifier service + Caddy) runs on one AWS `t4g.medium` with 30 GB.

6. Running a node today

Local devnet (the canonical recipe, `identity-chain/docs/protocol/devnet-runbook.md`):

```
go run ./cmd/idchaind init devnode --chain-id mintid-devnet-1 --home .devnet
go run ./cmd/idchaind keys add devval --keyring-backend test --home .devnet
go run ./cmd/idchaind genesis add-genesis-account devval 1000000000stake --keyring-backend test --h
go run ./cmd/idchaind genesis gentx devval 100000000stake --chain-id mintid-devnet-1 --keyring-back
go run ./cmd/idchaind genesis collect-gentxs --home .devnet
go run ./cmd/idchaind genesis validate --home .devnet
go run ./cmd/idchaind start --home .devnet
go run ./cmd/idchaind query identityparams params --home .devnet   # expect presentation_lifetime_s
```

Shortcuts: `make devnet-init | devnet-start | devnet-query | devnet-clean`. The binary builds reproducibly – `make build-reproducible` builds `idchaind` twice in Docker and asserts identical checksums.

The internal testnet (MINT-212):

```
Internet ──443──> caddy ──┬──> 127.0.0.1:26657  idchaind RPC   (TLS, internal endpoint)
         ──26656─> idchaind p2p                └─ single validator, mintid-testnet-1
                          └──> 127.0.0.1:8471   verifier service (TLS, internal endpoint)
```

Brought up and operated from the repo root: `make testnet-artifacts` → `testnet-home` → `testnet-launch` → `testnet-deploy` → `testnet-smoke`; teardown with `make testnet-down`. Day-2: `journalctl -u idchaind -f`; `sudo systemctl restart idchaind mintid-verifier caddy`. Redeploys never wipe chain state. The systemd unit is minimal: `idchaind start --home /opt/mintid/home`, restart-on-failure, `LimitNOFILE=65536`. Public ports: 443 (RPC via TLS), 26656 (p2p); RPC/gRPC/REST bind loopback.

Honest gaps (explicit, by design): single validator only – no sentry topology, failover untested until the multi-node testnet (V0-1); proof engine is the fail-closed pre-F3 stub; `x/selfrevoke` is an empty directory; council genesis is empty and inert; issuer registry empty with all fees at 0; flow A has neither params nor messages; no validator runbook, signed releases, snapshots or remote-signer setup yet. Footnote: the deployed testnet genesis still shows the SDK's default 13% inflation params – dead code; the chain mints solely through the custom `x/fee` MintFn.

7. Key management

- **Separate consensus and operational keys**; sentry-node topology; remote/HSM-backed signing where possible; monitoring; state snapshots; reproducible builds (SPEC §5.3). HSM support for validator signing keys is a should-have — but never callable during block execution; remote signing sits outside the state machine.
- **Consensus key**: ed25519, at `<home>/config/priv_validator_key.json` with its state file. Compromise is a §5.3 incident: emergency rotation, possible jail pending evidence.
- **Operator key**: secp256k1 in a keyring. The `test` backend is devnet/testnet-only — never for real funds.
- **Independence is contractual**: bootstrap service agreements must not grant anyone else control over your keys or votes; that independence is what qualifies your stake for the launch-gate arithmetic.
- You never custody holder keys (R3) — nothing in the role ever brings you near user secrets.

8. Governance: where you have power and where you don't

BODY	DECIDES	YOUR ROLE
`x/gov` **(token-weighted)**	Everything that changes software or parameters: upgrades, consensus params, `x/identityparams`, council size/threshold, cut points within ratified bands	**Full** – your stake and your delegators vote
`x/council` **(threshold vote, ~5-of-9)**	Operational gates: issuer/verifier admission, suspension, bond adjudication, treasury co-signing	**None** – not stake-weighted; the council never changes software
`x/treasurypolicy`	External-reserve moves (council + independent custodian co-signature)	**None** – and validator rewards are a prohibited use of the reserve

Early-network duties the spec assigns to governance: keep the validator cap conservative until bonded stake reaches the simulation's self-sufficiency threshold, front-weight early emission toward validators, and delay any feature whose compromise is irreversible. Expect governance drills (issuer suspension, key rotation, reserve denial) as milestone exit criteria, and continuous monitoring of concentration, liveness and equivocation post-launch.

Launch gate (§5.4): mainnet requires four measurable thresholds – bonded % of cap, independent operator count, Nakamoto-coefficient floor, and a fiat attack-cost proxy – all set by the Phase-0 simulation. Staging (ADR-0007) narrows what launches first (identity-only MVP; escrow/disputes later on organic volume) but never relaxes this gate. No mainnet without independent application, cryptography, infrastructure **and economic** audits (R20).

Annex – Requirement quick reference

ID	ONE LINE
R1	Sovereign PoS: permissionless admission, bounded active set, staking, delegation, slashing, predictable upgrades.
R2	The state you replicate never contains PII, payloads or presentation logs.
R3	You must never custody holder private keys.
R9	Issuer heartbeats every ≤30 s – the dominant recurring transaction load.
R10	Emergency revocation is effective on finality – your finality latency is the network's revocation latency.
R17	Hard cap, published declining emission, burns; implemented as the `x/fee` MintFn.
R19	State scales with issuers/verifiers/validators – never with holders or presentations.
R20	No mainnet before independent audits, attack exercises, bug bounty, IR plan.
R26	Paid off-chain services never influence consensus rewards – the firewall.
R27	Ordinary validators prune; archival nodes keep history and evidence commitments.

Manual v0.1 (public edition) – 2026-08-09. Sources: `master/SPEC-1` *v2.0 (§5, §6, §7.0, §9.3, §15–16), ADR-0004…0009,* `master/Genesis-Allocation-and-Transparency-Policy.md`, `docs/economics/` *draft report,* `deploy/testnet/` *(MINT-212),* `identity-chain/docs/protocol/devnet-runbook.md`. *All economic figures are illustrative bands pending the external simulation; testnet parameters are SDK defaults, not ratified values.*