Replicas — what they are, what they do, who runs them, what they earn

---

Date: 2026-05-25 **Status:** Detailed explainer on the replica role + a revision pass on bond economics from `tokenomics-option-e-detail-2026-05-25.md`. Reflects two design changes James authorized 2026-05-25: (1) replica unbond period reduced from 180 → 90 days; (2) replica reward source is the open design question this document resolves. **Scope:** Plain-English explanation of the replica role; technical detail on snapshot signing, anti-entropy gossip, quorum design; reward-source comparison (token emissions vs XMR fee share vs hybrid) with recommendation; updated bond + slashing parameters; user personas across operator categories.

> ■■ ***Alignment note (2026-06-28) — read first.*** *This is a dated iteration record, largely superseded by `replica-system-2026-05-25.md` and the whitepaper (§9, §11). Two later decisions change parts of it; canonical source: `DECISIONS.md` (D-013). -* ***The 5,000 XMR mint-fee cap was removed (D-013).*** *The post-cap reward-continuity analysis here ("scalable past the 5,000 XMR cap," "capped by the 5,000 XMR cap," etc.) is moot — fees now flow continuously, uncapped. -* ***Epoch is now two-cadence:*** *registry epoch = 7 days (snapshots), attestation epoch ≈ 1 hour (emission). This doc's "epoch ≈ one hour" snapshot-building cadence is superseded — snapshots are weekly.* ***Canonical R■ = 1,000 token per attestation epoch; this doc's "5,000 token per epoch" figure is superseded.***

---

0. Why this document exists

In conversation 2026-05-25 James flagged that the replica role hadn't been clearly enough explained in `tokenomics-option-e-detail-2026-05-25.md` to make an informed call on the bond economics. This document fills that gap.

It also resolves two open design parameters by James's direction:

1. **Unbond cooling period: 180 days → 90 days.** Faster capital recovery for operators; trade-off analyzed in §12.4.
2. **Reward source: open between (a) token emissions and (b) % of agent-registration fees in XMR.** Comparative analysis + recommendation in §11.

Like the prior documents, this is NOT a binding decision. It's a depth pass that makes the design concrete enough to evaluate against alternatives.

Part 1 — Plain English

1. What a replica is, in one paragraph

A **replica** is a server somebody operates that keeps a verifiable, signed copy of the public list of every Nightshade-registered AI agent. The list itself is called the **registry**. Multiple replicas exist on purpose: if there were only one, that one party could lie about who was registered (add fake agents, hide real ones, change capability descriptions). By having five replicas each independently signing the same list, anyone using Nightshade can check that at least three out of five agree before trusting the list. **A replica is the trust anchor of the agent directory.** Without replicas, agents can't find each other safely.

2. The registry, and why it needs to be replicated

When Bob (from the earlier persona documents) registers his translation agent, what actually happens technically is that the mint authority signs a blind credential, and then Bob submits a request to be **listed in the registry**. The listing contains his agent's verifying key, the capability the agent provides ("translation"), the price ("0.001 XMR per call"), the endpoint where the agent is reachable, and validity dates.

Now suppose Carol's agent wants to find a translator. Her agent queries the registry, gets Bob's listing back, verifies the cryptographic signatures, and proceeds to the §22 handshake to hire Bob's translator. **The integrity of that whole flow depends on Carol being able to trust the registry's answer.**

If the registry were just a single server, then:

- The operator of that server could **add fake agents** that don't really exist, taking payments and never delivering services.
- They could **hide real agents** from competitors, forcing Carol to pick a specific provider.
- They could **change capability descriptions or prices** silently between when Bob registered and when Carol queries.
- They could **be coerced by a government** to censor specific agents or operators.
- They could **be hacked** and the attacker could do all of the above.

By having multiple replicas each independently signing the same list, none of these attacks works unless the attacker compromises a majority of the replicas. **That's why the registry is replicated.** It's not for performance or scaling — it's for trust distribution.

3. What a replica does, day to day

In any 24-hour period, a replica server is doing about a half-dozen things continuously:

- **Accepting submissions from operators.** When Bob registers his agent, his SDK sends the registration to each of the five replicas over Tor. Each replica independently verifies that Bob's submission is well-formed (correct canonical encoding, valid signatures, valid mint credential).

- **Talking to peer replicas via gossip.** Every 30 seconds, each replica picks one of the other four at random and runs an anti-entropy reconciliation: "what registrations do you have that I don't have, and vice versa." This makes sure all five replicas converge to the same view of pending registrations within ~60 seconds of any new submission.
- **Building snapshots at epoch boundaries.** An "epoch" is a fixed time period (provisionally one hour). At the start of each epoch, every replica independently looks at the set of registrations + revocations it has accepted up to the previous epoch's end, sorts them canonically, and builds a `SnapshotBody`. **Because the input set is deterministic (everyone has gossip-converged) and the construction is deterministic, all five replicas independently produce the bit-identical same `SnapshotBody`** — they don't need to coordinate to agree.
- **Signing the snapshot.** Each replica signs the snapshot's hash with its own Ed25519 key. Now there's a `SignedSnapshot` carrying five signatures (or three, if two replicas are offline).
- **Publishing the signed snapshot.** Clients pull snapshots from any replica; verification is local. The 3-of-5 quorum check is what makes the snapshot trustworthy.
- **Handling emergency revocations.** If an operator's agent key gets compromised, they can issue an emergency revocation that doesn't wait for the next epoch's snapshot. All replicas need to sign an `EmergencyOverlay` quickly (within 5 minutes per spec §24.4). This is the "we need to revoke this agent NOW" mechanism.
- **Maintaining a transparency log.** Per spec §19.3, each replica publishes an append-only chain of `TransparencyLogEntry` records — basically "I attested snapshot X at time Y." This is what makes split-view attacks detectable. If replica A serves snapshot X to Bob but snapshot X' to Carol, the transparency logs reveal the divergence.

That's the protocol-level work. Operationally on top of that, the replica operator handles: server uptime, monitoring, Tor onion-service configuration, SDK-pinned public key management, sled-based persistence for crash recovery, log rotation, software updates.

4. Why five replicas, not one or ten

Three numbers matter: the quorum threshold (3), the replica set size (5), and how many can fail before the protocol breaks (2).

Why three for the quorum? Three is the minimum that lets a client verify "this snapshot wasn't fabricated by a single rogue operator." If the threshold were two, two corrupt operators could sign a fake snapshot and clients would have to accept it. Three means an attacker has to corrupt three different operators to forge a snapshot — a much higher bar.

Why five for the set size? Five is small enough to actually recruit, onboard, and trust-evaluate the operators. It's large enough that no single operator failure (server crashes, operator leaves, operator gets compromised) breaks the system — the quorum still has four other replicas to draw three signatures from. **Two replicas can fail before the quorum becomes unmeetable.** That's a meaningful safety margin without requiring an army of operators.

Five also makes the team-vs-external split work cleanly: 2 team-operated + 3 external = 5. If only the team operated, the protocol's trust would entirely depend on the team — Carol would have to trust the team not to add fake agents or censor. If only externals operated, the team would have no operational visibility into the system it built. Two team + three external means: team can't unilaterally corrupt (they need at least one external co-conspirator), externals can't unilaterally corrupt (they need at least one team co-conspirator), and the team has direct ops visibility through their own two nodes.

Why not three or seven? Three is unforgiving — any single failure breaks the quorum. Seven is more redundant but harder to recruit. Five is the established privacy-tech sweet spot: certificate transparency uses similar quorum structures; many sigstore deployments use 5–7 trusted root keys.

5. Who actually runs replicas

Five concrete examples of who runs replicas. These aren't real people; they're composite personas representing real categories.

5.1 Alice — sysadmin running a replica from a colocated server in Berlin

Alice has been operating Tor relays and Monero remote nodes for years as a paid hobby. She runs one of the three external Nightshade replicas. She bonds tokens to participate; she earns rewards (paid in token + a small XMR share — see §11). Her replica runs in a rack in a Berlin colocation facility she shares with three other privacy-tech projects. She monitors uptime via Prometheus, runs the replica binary on Linux, and reads the protocol-update notes for each release. **She has no business relationship with the team beyond the protocol's economic mechanism.**

5.2 Wei — runs a managed hosting service that hosts replicas for several privacy projects

Wei operates a small DevOps consultancy that specializes in privacy-oriented protocol infrastructure. He hosts replicas for Nightshade, Tor exit nodes for another project, and Monero supernodes for a third. He runs **two Nightshade replicas** — one on infrastructure he owns, one on a Hetzner box — under different signing keys so the protocol sees them as distinct operators. The Foundation's KYC-light verification process (modeled on Hyperliquid's Delegation Program but simpler) confirmed his two replicas are operationally independent. Wei is in this for the recurring rewards; he doesn't need to philosophically care about privacy, just provide reliable infrastructure.

5.3 Privacy-tech university lab — runs a replica as part of an applied-cryptography research program

A research group at TU Delft runs one of the replicas as part of a privacy-infrastructure-and-economics research program. The lab director uses the replica's operational data (anonymized) to publish papers on practical Byzantine-fault-tolerance patterns in adversarial settings. Their bond is paid for by the lab's research budget; the rewards offset operating costs. The lab's incentive is research output + grad student employment, not direct revenue. **They explicitly chose Nightshade over other replicated systems because the role is well-bounded and academically interesting.**

5.4 Privacy-conscious corporate (think Brave / Mullvad / Proton) — runs a replica as part of its brand-aligned infrastructure

A privacy-focused corporate (let's call them "Mullvad-style") operates a replica as part of their broader "we run privacy infrastructure" public commitment. Their bond is funded from operating budget; rewards offset infrastructure costs but aren't the goal. Their incentive is brand alignment with their user base. They publicly document their replica's existence as part of their privacy-infrastructure portfolio.

5.5 The Foundation — runs two team-operated replicas

Nightshade Foundation (the DAO-governed entity from the Option D structure in the prior doc) directly operates two replicas. These are funded from the Foundation budget, not from operator economics. Their existence is mostly for two reasons: (a) ensures the protocol has team operational visibility through nodes the team directly monitors and patches; (b) provides quorum redundancy if external operators have correlated downtime (data center outage, software bug, coordinated attack on the broader external operator set).

6. What replicas earn

Two streams (see §11 for the full design comparison):

1. **A small per-epoch XMR share** — taken from the agent-registration fee pool. Provisionally 1% of each 0.027182 XMR mint fee divided proportionally among the replicas that signed the snapshot containing that registration. Per registration: ~0.000272 XMR ≈ $0.05 at current prices, divided across however many replicas signed (typically 3–5). At 100 registrations/month with 5 active signers, each replica earns ~0.0054 XMR/month ≈ $1.10. Small. Bootstrap-only meaningful at scale.
2. **Token emissions** — from the protocol's bootstrap reserve, mintage rate calibrated against marketplace burn per the HIP-138 lesson. Provisionally 5,000 token per epoch per active replica during the first 12 months, decaying linearly over 36 months to a floor of 500 token/epoch. At current notional 1 TOKEN = 0.0001 XMR, that's 0.5 XMR/epoch ≈ $100/month per replica during bootstrap.

Combined provisional bootstrap economics: an active replica earns roughly $100-$200/month at v1 launch (mostly from token emissions, secondarily from XMR fee share). At maturity (post-bootstrap), economics depend on registration volume + token price growth. **This is small money in absolute terms but appropriate for what is, fundamentally, a low-touch infrastructure role with a long-tail revenue curve.** Replica operators aren't getting rich; they're being compensated for service rendered + token upside.

6.1 Unbond cooling: 90 days (revised from 180)

When an operator decides to stop running a replica, their bond doesn't release immediately. There's a cooling period during which the bond remains slashable. **Originally proposed at 180 days; James reduced to 90 days as of 2026-05-25.** Trade-off: shorter cooling means faster capital recovery for operators (good for recruitment), but shorter post-exit detection window for misbehavior discovered after the operator has left (slightly worse for protocol safety). 90 days strikes a reasonable balance — long enough for any audit-driven discovery of fee/snapshot inconsistencies, short enough that operators don't feel their capital is locked up for half a year.

Part 2 — Technical detail

7. The protocol role — spec §19

The replica role is normatively defined across spec §18–§19 (snapshot construction + quorum) and §24 (emergency overlay) and §19.3 (transparency log). The reference implementation is `nightshade-replica` (Phase 5; slice 1 landed 2026-05-25; slices 2–8 pending).

The replica's protocol responsibilities, mapped to specific spec sections:

Responsibility	Spec section	Implementation slice
Accept operator submissions	§17.6 (registration) + §23.1 (revocation)	nightshade-replica slice 2
Validate canonical encoding + signatures	§5–§9 + §14.3	nightshade-replica slice 2
Build per-epoch `SnapshotBody`	§18.4	nightshade-replica slice 3
Sign with replica key	§19.1	nightshade-replica slice 3
Gossip with peer replicas	§19 (under-specified; Q2 resolution 2026-05-25 = anti-entropy + epoch-batched deterministic construction)	nightshade-replica slice 4
Receive + sign emergency overlay	§24.3	nightshade-replica slice 5
Maintain transparency log	§19.3 (audit fix H3)	nightshade-replica slice 6
Persist state for crash recovery	(impl)	nightshade-replica slice 7 (sled per Q1 resolution)
Multi-replica testnet integration	(operational)	nightshade-replica slice 8

8. What gets signed

Four distinct signed artifacts. All four use Ed25519-strict (per spec §2.1) over canonical-encoded body bytes hashed under domain-separated tags.

8.1 `SignedSnapshot` (per epoch)

```
SnapshotBody {
   epoch: u32,
   published_at_unix_seconds: u64,
   registry_challenge: [u8; 32],
   agent_entries: Vec<AgentEntry>,  // canonically sorted
   revocations: Vec<RevocationEntry>,  // sorted by revocation_tag
}

snapshot_hash = SHA-256("nightshade-snapshot-v1" || 0x00 || canonical(SnapshotBody))

SignedSnapshot {
   body_bytes: Vec<u8>,  // canonical encoding of SnapshotBody, verbatim
   replica_signatures: Vec<ReplicaSignature>,  // sorted by replica_id
}

ReplicaSignature {
   replica_id: [u8; 32],
   signature_bytes: Vec<u8>,  // Ed25519 signature on snapshot_hash
}
```

Verification: `verify_quorum(replicas: &[(replica_id, pk)])` checks ≥3 distinct replica signatures verify against `snapshot_hash` using strict Ed25519. Unknown replicas silently skip (don't count toward quorum but don't fail it). This is the workspace's `nightshade-registry` slice 2 implementation, already verified.

8.2 `SignedEmergencyOverlay` (per refresh, typically every ~60 seconds)

```
EmergencyOverlayBody {
   as_of_unix_seconds: u64,
   valid_for_seconds: u64,  // = 300 per §24.3 v1
   entries: Vec<EmergencyRevocation>,  // sorted by revocation_tag
}

overlay_hash = SHA-256("nightshade-emergency-overlay-v1" || 0x00 || canonical(body))
```

Same 3-of-5 quorum. Clients check `now - as_of_unix_seconds < valid_for_seconds + grace`; if overlay is stale, fail-closed per spec §24.4. Implementation: registry slice 5a (already verified).

8.3 `SignedTransparencyLogEntry` (per replica per published snapshot)

```
TransparencyLogEntryBody {
   replica_id: [u8; 32],
   epoch: u32,
   snapshot_hash: [u8; 32],
   published_at_unix_seconds: u64,
   prev_log_hash: [u8; 32],  // first entry: zero
}

entry_hash = SHA-256("nightshade-transparency-v1" || 0x00 || canonical(body))

SignedTransparencyLogEntry {
   body_bytes: Vec<u8>,
   signature_bytes: Vec<u8>,  // Ed25519 on entry_hash
}
```

Single replica signs each entry (not a quorum). Clients aggregate ≥3 distinct replicas attesting the same `snapshot_hash` at the same epoch via `ReplicaQuorumTransparencyCheck`. Audit fix H3 enforcement point. Implementation: registry slices 6a + 6b (already verified).

8.4 Why these four artifacts and not e.g. per-registration signatures

In an alternative design, each replica could sign each individual registration as it's accepted. This was considered and rejected: per-registration signing creates O(N) signatures per epoch (where N = registrations per epoch), while snapshot-based signing is O(1) per epoch. The snapshot-based design also makes clients' verification O(log N) for lookups + O(1) for trust establishment, vs O(N) for both in the per-registration design. The trade-off is that revocations need to wait until the next epoch boundary to land in a snapshot — which is why the emergency overlay exists.

9. The 3-of-5 quorum

9.1 Why these specific numbers

Byzantine fault tolerance theory says: to tolerate f Byzantine faults, you need 3f+1 honest participants. For f=2 (the minimum useful tolerance — 0 means no protocol redundancy, 1 means any single corruption breaks safety), you need 7 participants. **But** that's for full BFT consensus, where participants need to actually agree on a sequence of operations.

Nightshade's replica problem is much simpler than BFT consensus. The deterministic snapshot construction means all honest replicas independently produce the bit-identical same `SnapshotBody` without coordinating. The quorum is purely about **attestation count for clients to trust the result**, not about reaching agreement among replicas. Reframed this way, the threshold is a security parameter, not a consensus parameter.

For attestation, a 3-of-5 quorum gives:

- **Sybil resistance:** an attacker needs to control 3 replicas to forge a snapshot. With per-replica bonds slashed at 100% for double-signing, that's an explicit economic cost.
- **Liveness:** 2 replicas can fail without breaking the quorum.
- **Operational tractability:** 5 operators is a small enough set to vet, onboard, monitor, and replace.

9.2 Why not 5-of-5

5-of-5 would mean any single operator failure breaks the protocol. Unworkable for liveness — even routine maintenance windows would cause outages.

9.3 Why not 4-of-7

7 replicas means recruiting + vetting 7 operators. The team scale doesn't justify 7 at v1 launch; 5 is the minimum that satisfies the security parameter while staying recruitable. v2 candidate: expand to 7 or 9 once the operator economy is established.

9.4 What happens if a replica goes rogue

Three categories of replica misbehavior + the detection mechanism:

Misbehavior	Detection	Recovery
Replica A double-signs (signs two different `SnapshotBody`s for same epoch)	Transparency log shows two `SignedTransparencyLogEntry`s with same `(replica_id, epoch)` but different `snapshot_hash`. `ReplicaQuorumTransparencyCheck` surfaces the inconsistency.	Slash A's bond 100%. Other 4 replicas continue; 3-of-4 still meets quorum (3-of-5 with one replaced by zero). New replica recruited from the candidate pool.
Replica B refuses to accept a valid registration	Auditor's records of paid mint fees → §17.9 issuance log → snapshot. If a paid registration doesn't appear in B's snapshot but does appear in A/C/D/E's, B is provably refusing service.	Slash B's bond 50–100% based on pattern.
Replica C lags gossip — fails to converge within 2 epochs	Other replicas observe C's stale state via anti-entropy reconciliation. After grace period, replicas can vote to mark C inactive.	Slash 10–25%. C can re-sync and resume but loses rewards during stale period.

10. Anti-entropy gossip + deterministic snapshot construction

Per the Q2 resolution 2026-05-25, the gossip layer is **anti-entropy with epoch-batched deterministic snapshot construction**. Not Paxos-style consensus. Not Raft. The Cassandra/DynamoDB/Riak pattern.

Steps:

1. **Submission propagation (push).** When replica A accepts an operator submission, A POSTs it to all 4 peers via authenticated HTTP-over-Tor. Best-effort fire-and-forget; no ack required.
2. **Periodic anti-entropy reconciliation (pull).** Every `ANTI_ENTROPY_INTERVAL_SECONDS` (initial: 30s), each replica picks a uniformly random peer and runs a Merkle-tree-of-canonical-submission-identifiers reconciliation. The protocol surfaces "what do I have that you don't" + "what do you have that I don't" + exchanges the missing entries. Each entry is re-verified by the receiving replica (credential signature, mint signature, etc.).
3. **Epoch-batched deterministic construction.** At each epoch boundary, every replica deterministically builds its `SnapshotBody` from the canonical-encoded-and-sorted-and-deduplicated set of submissions it has independently credential-verified by `epoch_start - SUBMISSION_CUTOFF_SECONDS` (initial: 60s). The 60s cutoff window absorbs the 30s anti-entropy interval plus 30s safety margin. With high probability, every honest replica has the same submission set within the cutoff, so all 5 produce identical `SnapshotBody`s.

The 3-of-5 quorum IS the BFT-tolerance layer. The gossip layer doesn't need to also be BFT-consensus; that would be over-engineered. A malicious replica that fabricates an unauthorized registration is detected by the receiving replica's credential-signature verification. A malicious replica that censors a legitimate registration is detected by §21 auditor + cross-replica comparison.

11. Replica reward design — three options

This is the open design question James asked me to resolve. He proposed two options ("token OR % of fees for agents registering") and asked for analysis. I'm adding a third — the hybrid — because the strongest defense combines the strongest features of both.

11.1 Option I — Token emissions only

Mechanism: at each epoch boundary, after replicas have signed and published, the protocol mints N tokens to each replica that signed the snapshot. N is calibrated against marketplace burn per the BME equilibrium from `tokenomics-option-e-detail-2026-05-25.md` §7.4. Bootstrap subsidy structure: 5,000 token/epoch/replica at v1 launch, decaying linearly over 36 months to a 500 token/epoch floor.

Pros: - **Clean utility framing.** Replicas earn token for service rendered; the token has independent utility (bond, marketplace credits). This is the Helium / Filecoin block-rewards pattern — neither has drawn SEC enforcement. - **Decoupled from mint fee economics.** The existing PROJECT_OVERVIEW.md §7 fee distribution (65% / 15% / 10% / 10%) can stay as-written or be rewritten on its own timeline; replica rewards don't depend on the XMR pool allocation. - **Scalable past the 5,000 XMR cap.** Mint fee revenue stops accumulating once the cap is hit; if replicas depended on XMR fee share, post-cap economics break. Token

emissions continue regardless. - **Predictable for operators.** Operators can model their expected token revenue using the published emission schedule. No "did anybody register this epoch?" volatility.

Cons: - **Token-price volatility.** If token price crashes, replica revenue (in USD terms) crashes. Operators may exit. - **Dilutive if marketplace burn fails to materialize.** The HIP-138 trap from §7.4 — but the `BOOTSTRAP_EMISSION_CAP` and sunset clause are explicit defenses. - **No connection between replica work and economic flows from registrations.** Philosophically, a replica that signs a snapshot containing 1,000 registrations gets the same reward as one signing a snapshot containing 1. The number of registrations doesn't reward the replicas processing them.

11.2 Option II — XMR fee share only

Mechanism: P% of each agent-registration mint fee (0.027182 XMR) is set aside in a "replica reward pool" rather than flowing through the existing §7 fee distribution. The pool is split per-registration among the replicas that signed the snapshot containing that registration. Provisional P = 1%, so per-registration replica reward = 0.000272 XMR, divided across however many replicas signed (typically 3–5 of 5).

Pros: - **Real revenue, no inflation.** Replicas earn from people who actually paid for service. No emission needed. - **Direct work-to-reward coupling.** More registrations in your snapshot = more revenue. Aligns operator incentives with registration throughput. - **Stable purchasing power.** XMR-denominated reward; not subject to token-price volatility. - **Funding source already exists.** The mint fee pool is real; allocation can be reorganized within counsel's review of §7.

Cons: - **Securities-exposure profile is closest to dYdX v4** of any option in this design. "Mint fees paid by third parties → bonded operators receive fees" maps almost exactly to "trading fees paid by traders → token stakers receive fees." The mitigation: the recipients are **bonded operators doing real infrastructure work**, not passive token stakers. Filecoin SPs receive deal fees + block rewards without enforcement because they're doing storage work. This framing is defensible but counsel-sensitive. - **Capped by the 5,000 XMR cap.** Post-cap, no new fee revenue arrives; replica rewards must transition to a different model. **This is structural.** - **Volatile per-replica revenue.** Slow registration weeks → low rewards. Replicas need cash-flow stability or operator retention suffers. - **Reorganizes §7.** Either takes from existing recipients (65% / 15% / 10% / 10%) or fills the missing 10%. Counsel review.

11.3 Option III — Hybrid (recommended)

Mechanism: Both streams. Token emissions provide baseline economic stability (predictable + decoupled from registration volume); XMR fee share provides marginal pay-per-work alignment + bridges to post-cap economics through a real cash flow.

Concrete provisional parameters: - Token emissions: 5,000 token/epoch/replica at launch → 500 token/epoch/replica floor at month 36 (same as Option I). - XMR fee share: 1% of each registration mint fee → replica reward pool → split per-registration among signing replicas (same as Option II). - Both subject to bond requirement (slashing applies to both). - Both stop on operator exit + 90-day cooling.

Why the hybrid:

- **Stability + alignment.** Token emissions give operators baseline income (essential for recruitment); XMR share rewards the work of processing actual registrations (philosophical alignment +

counsel-defensible utility framing).

- **Pre-cap and post-cap continuity.** XMR share works pre-cap; token emissions continue post-cap. The hybrid doesn't break at 5,000 XMR.
- **Legal defensibility profile better than either pure option.** XMR-only is closer to dYdX v4 (the worst profile); token-only is closer to Helium block rewards (cleaner). The hybrid sits between but, critically, **routes the bulk of replica reward through token emissions (the cleaner pattern) and uses XMR share only for marginal pay-per-work**. A counsel review can affirm "the replica's primary compensation is token emissions for service rendered; the small XMR component reflects per-unit work and is structurally bounded by the registration cap."

Recommended choice: Option III.

11.4 Decision dependency

Option choice is **counsel-reviewable**. The differences matter to the securities-law analysis. James can override; this document presents the analysis and recommendation. The current PROJECT_OVERVIEW.md §7 fee-distribution writeup will need a parallel rewrite (independent of which option is chosen here) to remove the "early holders / buybacks" language flagged in the legal-surface session. The §7 rewrite + this option choice are best done together with counsel.

12. Bond + slashing (revised)

12.1 Bond size (unchanged from prior doc)

Role	Provisional bond	Rationale
Replica operator	100,000 token	Aligned with the lower end of validator bonds in protocols with similar trust roles (Hyperliquid's 10k HYPE self-bond is the minimum; we want higher to deter Sybil attacks given a smaller active set).
Mint authority	1,000,000 token (10× a replica)	Higher trust position → higher slashing exposure → higher bond.

12.2 Slashing conditions (unchanged)

For replicas:

- Double-signing: 100% bond slash; permanent operator ban.
- Refusal of valid registrations: 50% on first instance; 100% on second.
- Operating with stale state >2 epochs behind gossip: 10–25%.

- Late refresh of emergency overlay beyond §24.4 5-minute target: 5–15%.

For mint:

- Double-issuance: 100%.
- RSA key compromise: 100%.
- Late publication of issuance log: 5–20% based on staleness.
- Refusal of valid blinded submissions: 50–100%.

12.3 Unbond cooling: 90 days (revised from 180)

As of 2026-05-25 per James, cooling period reduced 180 → 90 days.

The cooling period exists to keep the bond slashable for some time after the operator says "I want out." During cooling, the operator must continue to operate honestly. Misbehavior detected during cooling is still slashable against the unbonding balance.

Why 90 days is defensible:

- 90 days is long enough for the §21 auditor's quarterly cycle to detect any late-discovered mint-fee-versus-snapshot inconsistencies that surfaced after the operator announced exit.
- 90 days is short enough that operators don't feel their capital is locked for half a year (recruiting concern).
- Comparable to Cosmos validator unbond (21 days; aggressive) and Polkadot validator unbond (28 days; aggressive); longer than both, reflecting the protocol's slower audit cadence. Shorter than Filecoin's sector-collateral release (180+ days; conservative for storage proofs).
- The Cosmos and Polkadot models work because misbehavior detection is fast (on-chain proof immediate). Nightshade's detection cadence is mostly slower (auditor cycle + cross-replica reconciliation). 90 days balances the two.

What we lose at 90 vs 180:

- A specific risk: an operator who has been gradually misbehaving in subtle ways might exit, complete 90-day cooling, and be paid out before a comprehensive audit catches the pattern. At 180 days, that audit has more time to surface.
- Mitigation: faster on-chain misbehavior detection (the transparency-log check makes double-signs immediately detectable); auditor's continuous monitoring (not just quarterly) catches refusal-of-service patterns within the cooling window.

12.4 Slashing destination

Slashed tokens are **burned** (sent to verifiable burn address). Not redistributed to other operators (creates perverse "false flag" incentive). Not sent to the Foundation treasury (creates conflict-of-interest if Foundation is the slashing arbiter). Burn is the cleanest.

13. Failure modes specific to replicas

13.1 Correlated downtime

If a popular hosting provider has an outage and multiple external replicas are co-hosted there, the quorum can be threatened. With 5 replicas, 2 simultaneous failures still leave 3-of-5 (quorum met but no fault tolerance margin); 3 simultaneous failures break quorum.

Mitigation: Foundation's KYC-light verification of new replica operators includes a hosting-diversity check. The Foundation's own 2 replicas are run on different infrastructure than typical external choices.

13.2 Coordinated attack on 3 replicas

If an attacker compromises 3 of 5 replica signing keys, they can forge snapshots. This is the catastrophic failure mode for the registry's trust model.

Mitigation: - Operator KYC-light gating raises the bar for attackers to know who to target. - Geographic + jurisdictional diversity of operators (Berlin, US, EU, Asia distribution). - Hardware security modules for signing keys (recommended; not strictly required at v1 launch but the spec for the eventual operator onboarding doc includes this). - The transparency log + auditor would detect the forgery within minutes-to-hours; emergency overlay can quickly revoke any rogue agent the attacker registered.

13.3 Cooling-period attack

Operator exits, completes 90-day cooling, gets paid out. 95 days after their exit, an audit surfaces misbehavior they committed during their active period.

Mitigation: - 90-day cooling is calibrated to the auditor's monitoring cycle; in practice the auditor is monitoring continuously, not quarterly. - Slashable evidence must be on-chain (transparency log inconsistencies) or auditor-attested (issuance log vs registration log inconsistencies). Both are visible and accumulate during the active period. - If post-cooling-discovered misbehavior becomes a real problem, the cooling period can be extended via Foundation governance vote.

13.4 Reward economics insufficient to maintain operators

If real registration volume + marketplace burn doesn't materialize as expected, the combined token + XMR replica reward may be insufficient to retain operators. Operators exit; quorum threatened.

Mitigation: Foundation budget includes a contingency to pay direct subsidies (in XMR or token) to retain replicas during slow growth periods. The reward economics in §11 are bootstrap-calibrated to be deliberately attractive during the first 12 months when actual revenue is uncertain.

13.5 Replica software bug

If a bug in `nightshade-replica` causes a specific replica to sign incorrect snapshots, the transparency log would surface the divergence, but the operator wasn't malicious. Slashing a buggy honest operator is unfair.

Mitigation: - Foundation governance can vote to refund a slash if subsequent post-mortem proves the cause was a software bug rather than malicious intent. - Reference implementation undergoes the standard protocol's review process before each release (currently James + me; eventually community review + ideally a paid audit before mainnet per the §14 pre-mainnet gate).

14. Open design questions specific to replicas

1. **Epoch duration.** Provisional 1 hour. Shorter = faster registration→snapshot latency but more compute load per epoch. Longer = lower compute load but registrations wait longer to appear.
2. **`ANTI_ENTROPY_INTERVAL_SECONDS`.** Provisional 30s. Trade-off: shorter = faster gossip convergence + more network chatter; longer = slower convergence + lower bandwidth.
3. **`SUBMISSION_CUTOFF_SECONDS`.** Provisional 60s. Must exceed `ANTI_ENTROPY_INTERVAL_SECONDS` by a comfortable margin (say 2×) to ensure all honest replicas converge before the cutoff.
4. **Replica reward source choice.** Option I / II / III per §11. Recommended: III. **Counsel-reviewable.**
5. **Bond denomination adjustment mechanism.** If token price spikes, fixed 100k token bond may price out new operators. Foundation governance vote to adjust target USD-equivalent value?
6. **Hardware security module requirement.** Currently recommended, not mandated. v2 candidate: mandate HSM for new replica onboarding.
7. **Foundation's own 2 replicas: who specifically runs them within the team?** Operational ownership matters for security; needs to be specified in the operator onboarding doc.
8. **Replica voluntary retirement vs forced removal.** If an operator becomes uncontactable, how does the protocol clean up? Foundation governance vote to recover the bond? Specifies in subsequent slice plans.
9. **Geographic + jurisdictional distribution policy for the 3 external operators.** No two from the same hosting provider; no concentration in any one legal jurisdiction beyond a threshold.
10. **Transition path for the 5-replica → potentially larger set at v2.** How is a new replica added mid-protocol? Bonded + Foundation-vote-confirmed + 1-epoch ramp-in?

---

15. Summary

Question	Plain English answer	Technical answer
What is a replica?	A server that keeps a verifiable, signed copy of the agent registry.	One of 5 nodes that signs `SignedSnapshot`, `SignedEmergencyOverlay`, `SignedTransparencyLogEntry` per spec §18–§19, §24.
Why 5?	2 team-operated + 3 external = enough redundancy + diversity, small enough to recruit.	3-of-5 quorum tolerates 2 Byzantine faults; 5 = minimum set for that threshold.
Why do operators run them?	Earn rewards in token + XMR share; bond their stake as skin-in-the-game.	Bond 100k token; earn token emissions + XMR fee share (Hybrid Option III recommended).
What stops a replica from cheating?	If they sign two different snapshots, anyone can see it. They lose their bond.	Transparency log + 3-of-5 quorum check via `ReplicaQuorumTransparencyCheck`. Slash 100% bond on double-sign.
How long is the unbond?	90 days (reduced from 180 per James 2026-05-25).	90-day cooling period during which bond remains slashable; balances operator capital recovery + late-audit window.
What if a replica goes offline?	The other 4 carry on; 3 signatures is still quorum.	3-of-5 means 2 failures tolerable without breaking quorum. Operator who's offline >2 epochs loses gossip-stale reward.
What if the team disappears?	The 3 external replicas keep the registry alive; the protocol still works.	Decentralized verification via SDK-pinned replica pubkeys; clients trust the quorum regardless of who's behind individual replicas.

Companion document to `tokenomics-research-2026-05-25.md` (the wide research) and `tokenomics-option-e-detail-2026-05-25.md` (the Option E first-pass design). Reflects the two design changes James authorized 2026-05-25. Neither this document nor either companion is a binding decision; together they make the design concrete enough that the decision can be made with counsel review.