ODNCA-TF-001 · Trust Framework

What you must trust, what you can verify, and what survives every failure

Status: v1.0 — Adopted

Companion documents: ODNCA-STD-001 (signed answers), ODNCA-STD-004 (commitments & certificates), ODNCA-STD-007 (verification), ODNCA-POL-003 (measures), ODNCA-GOV-001 (key stewardship).

1. The principle stack

Five principles, each one already normative somewhere in the standards — collected here because together they *are* the trust model:

1. **The chain is the root.** Validity and ownership are recomputable facts on a public blockchain. Every service reads the chain; no service replaces it.
2. **Central follows decentral.** Central components (registry, resolver, WHOIS) learn state from the chain via the fact layer — never the reverse. Where an administrative record and the chain disagree, the chain is right.
3. **Trust is optional at every level.** Every answer can be taken on trust, verified by signature, or proven against the chain (the three levels of ODNCA-STD-001 §9). Skipping verification is always the client's choice, never the protocol's requirement.
4. **Honesty over convenience.** Fallbacks are disclosed (`fallback: true`), propagation is admitted (`no_holder`), blocks say `blocked` with a reason, and nothing is ever dressed up as `not_registered` that isn't. An honest error is infrastructure; a comforting lie is debt.
5. **Permissionless base, recognition on top.** Nobody needs permission to claim, transfer, or build; recognition (verification, listing, TLD activation) adds services, never rights.

2. The three trust layers

Every interaction with a name touches up to three layers, each with its own proof:

Layer 1 — Transport: *is this answer authentic and fresh?*

Resolver answers are signed (fixed serialization, double-SHA256, secp256k1) over an exhaustively listed field set including the expiry — so an answer cannot be forged, altered in transit, or replayed after its TTL. Keys are pinned on first use and rotate only by announcement signed with the predecessor key (ODNCA-STD-001 §7). Transport trust never requires TLS, DNS, or any web2 chain of custody — the signature is the channel.

Layer 2 — Ownership: *does this name really belong to this holder?*

Three independent proofs, in ascending strength: the **signed answer** (the operator's attestation), the **outpoint check** (the chain's own word — verify `current.txid:vout` unspent at any node and you trust nobody), and the **ownership certificate** (a merkle path from the holder's leaf to an on-chain committed root, verifiable offline with the reference verifier, mintable as a permanent on-chain attestation; ODNCA-STD-004 §6). Registrar **verification** (the issuer-signed claim record, ODNCA-STD-007 §5) sits beside these as an administrative fact — it marks provenance, never validity.

Layer 3 — Content: *is what loads under this name what the holder published — and should it load?*

Content is addressed by outpoint: the name→content coupling names an exact inscription, and the txid form of `ord://` is immutable forever (ODNCA-STD-005 §4). What you load is what was inscribed — there is no server that can serve you something else under the same address. Whether it loads is the service question: measures per POL-003 apply at display time, honestly labelled, per operator, against a published signed blocklist.

3. The key inventory

All trust that is not pure chain math rests on published keys, each with one role and never shared across roles:

KEY	SIGNS	DEFINED IN
Resolver signing key	resolution answers	ODNCA-STD-001 §7
Registrar issuer key	claim records (name + recipient)	ODNCA-STD-007 §3.3
Commitment key	on-chain commit inscriptions	ODNCA-STD-004 §6
Proof-API key	certificate responses	ODNCA-STD-004 §6
Blocklist key	the published blocklist	ODNCA-POL-003 §6

One rotation procedure covers them all (successor signed by predecessor, published, anchored); stewardship duties in ODNCA-GOV-001 §8. Compromise of any single key is recoverable by rotation precisely because no key can rewrite the chain — the worst a stolen key can do is lie briefly, detectably, about facts anyone can recompute.

4. Escrow, solved by construction

Web2 registries deposit their data with third-party escrow agents so that a registry failure does not orphan the names. Here that protection is not a contract but a construction:

- **The chain is the deposit.** Every claim, transfer, and coupling-relevant fact is on a public blockchain; the full registry state is reconstructible by anyone running the published rules against public data.
- **The commitment chain is the receipt.** Periodic on-chain merkle roots — prev-linked into an auditable history, each binding the exact ruleset hash — prove what the state *was* at every committed height, compactly.
- **Certificates are the holder's copy.** Every holder can carry (and mint) their own offline-verifiable proof of ownership, needing no one's cooperation to present it.

Consequence: the failure of any operator — including the founding one — loses convenience, never names.

---

5. Failure modes, honestly

WHAT FAILS	WHAT HAPPENS	WHAT IS LOST
Registrar portal down	raw registration path remains (POL-002 §1); existing names unaffected	assisted convenience
Resolver down	any conformant resolver serves the same chain; run your own from the published rules	one operator's endpoint
Registry database lost	rebuilt from the chain via the indexer (owner-sync full mode)	administrative extras until resync
A signing key compromised	rotation per §3; stale lies fail the outpoint check	nothing durable
ODNCA itself disappears	rules, vectors, and commit history are published and anchored; anyone recomputes and continues	a coordinator, not a registry

The last row is the framework's real test, and its point: this ecosystem is designed so that its own authority is optional. *Don't trust us — recompute us* is not a slogan under the documents; it is the architecture in one line.
