ODNCA-STD-006 · Data Channels

Payment, mail, and content — three channels instead of a record model

Status: v1.0 — Adopted
Normative reference implementations: ORDnet SNS Resolver v1.1 (payment), ORDmail Core v4.1 (mail), ORDnet Registry v16.2x (content coupling, read surface), owner-sync v16.21 (state flow).

1. There is no record model — deliberately

DNS answers "what is coupled to this name?" with a managed table of record types. This ecosystem answers it differently: **a name has three data channels, and each channel's truth lives on the chain in its own native form.** There is nothing to administer that the chain does not already state:

CHANNEL	THE ON-CHAIN TRUTH	READ VIA
Payment	the current outpoint of the name's satoshi	signed resolver answer (ODNCA-STD-001)
Mail	1-sat message inscriptions owned by the recipient	the recipient's own wallet
Content	a 1Sat Ordinals inscription script in a sat	the name→content coupling (§4)

A "records" API in this world is a *reading window* onto these channels, not a management surface for record types. This document defines each channel's deployed form.

2. The payment channel

Fully defined in ODNCA-STD-001: address in (`name.tld` or `mailbox@name.tld`), signed answer out, `holder_script` is what you pay, the outpoint check is what you can prove. Nothing further is coupled or configured; payment truth *is* ownership truth.

Mailbox payment lookup. For mail-style addresses the service layer additionally exposes a mailbox lookup (`prefix@domain` → destination wallet and, when set, an encryption public key). An exact alias match answers with the alias wallet; a domain with catch-all enabled answers with the holder's wallet, honestly labelled `catch_all`. The chain-level invariant of ODNCA-STD-001 §5 always stands above this: whoever holds the sat holds every mailbox.

3. The mail channel

A mail message is not data *about* the recipient — it is property *of* the recipient:

1. **The message is a 1-sat inscription locked to the recipient's address.** The payload is the mail envelope, JSON: `{"p":"ordmail","op":"send","to","from","subject","body","encrypted","reply_to","attachments":[],"ts"}`. The recipient's inbox is, literally, a set of ordinals they own.
2. **A 1-sat OP_RETURN marker output** `["ORDnet.io","ordmail"]` travels in the same transaction as the discovery beacon — indexable without parsing every inscription on the chain.
3. **Attachments are separate 1-sat inscriptions** locked to the recipient (image, PDF, any MIME), referenced in the envelope's `attachments` array as `txid_vout` outpoints. Content and MIME live on chain; the envelope carries pointers.
4. **Encryption** is end-to-end: content keys (AES-256-GCM) sealed to the recipient's public key (ECIES), the public key discoverable via the mailbox lookup (§2). An encrypted envelope renders as "Encrypted message" to everyone but the key holder.
5. **Sats can ride along:** an ordinary P2PKH output to the recipient, appended after the fixed wire-format outputs so existing readers parse unchanged — money arrives on the same transaction as the message.
6. **Inbox protection is a client duty:** wallets MUST exclude 1-sat inscription ordinals from spendable coin selection, or a payment could silently consume received messages forever. (Reference behaviour: only bare P2PKH outputs above 1 sat are spendable money.)

4. The content channel

Content is a 1Sat Ordinals inscription — a script in a sat. What makes it *a name's website* is the **coupling**: the name's registry entry carries a target outpoint (`target_txid`, `target_vout`). A browser resolving `ord://name.tld` (ODNCA-STD-005 §4) reads the coupling and loads that inscription; updating the site means re-pointing the coupling to a new inscription, while every old version remains permanently addressable by its own txid.

Two refinements, both holder-managed and wallet-signed (§5):

- **Subdomains:** labels (`a–z0–9`, hyphen-internal, ≤63 chars, up to 50 per domain) each coupled to their own target outpoint — `blog.name.tld` points at its own inscription.
- **Routes:** per-subdomain path mappings (`subdomain` + `path` → outpoint) for multi-resource sites, pending the On-Chain Web standards for full site semantics.

The read surface is public: `GET /api/domain/<name>/records` returns the couplings (subdomains, routes, active listing if any) — the reading window of

§1. Content measures per ODNCA-POL-003 apply at *load* time in browsers, never at the coupling data itself.

---

5. Channel management: signed, never trusted

Every coupling mutation (set/remove target, subdomain, route, transfer, listing) is authorised by a **wallet signature over the operation and its arguments** — the registry verifies that the signer is the name's holder and never accepts a session or password as a substitute for the key. Premium-tier gating of some conveniences is registrar policy (out of scope per ODNCA-STD-003 §1); the signature requirement is not.

---

6. State flow: central follows decentral

The registry does not decide ownership; it *learns* it. The indexer — the fact layer — pushes state to the registry (owner and the origin/current outpoints per verified name) on a delta cadence with a full nightly reconciliation. The design rule is the sync's own first line: **"centraal volgt decentraal"** — central follows decentral. Two properties are normative:

- **The custody rule.** For names held in service custody, sync updates *outpoints only, never the owner* — the administered holder remains authoritative for `holder_script` while the chain remains authoritative for freshness. This is the deployed basis of the custody semantics in ODNCA-STD-001 §6.
- **Burned names are dropped** from sync (no current outpoint → nothing to serve), consistent with the burn classification of ODNCA-STD-001 §3.
