# ODNCA-STD-005 · URI Schemes

sns: for names and payments · ord:// for on-chain content

**Status:** v1.0 — Adopted

**Companion documents:** ODNCA-STD-001 (Resolution — normalization and signed answers), ODNCA-STD-003 (SNS-NAME-1), ODNCA-STD-006 (Data Channels, forthcoming), the On-Chain Web standards (forthcoming).

## 1. Why fix this now

Schemes are cheap to decide and expensive to repair: once links live in documents, QR codes, and apps, every ambiguity becomes permanent. This standard fixes two schemes and reserves one, matching what deployed software already accepts.

## 2. `sns:` — the name scheme (primary)

The canonical way to write an SNS name or mail address as a URI:

```
sns:<address>
address := [ <mailbox> "@" ] <name> "." <tld>

sns:alex.web3
sns:alexander@ordnet.web3
```

Rules:

1. **Syntax.** The part after `sns:` is exactly the address grammar of ODNCA-STD-001 §4.1; normalization is ODNCA-STD-001 §4.2 (which already strips the scheme prefix — deployed behaviour, now normative).
2. **Semantics.** An `sns:` URI identifies the *name*, not a specific action. The handling application chooses the action from context: a wallet opens payment to the resolved `holder_script`; a mail client opens composition to the mailbox; a browser MAY treat it as `ord://` on the same name (§4).
3. **No slashes.** `sns:` takes no `//` authority component: `sns:alex.web3`, never `sns://alex.web3`. Parsers SHOULD accept the sloppy double-slash form on input and strip it; emitters MUST NOT produce it.
4. **Emission.** Applications generating links, QR codes, or deep links for names MUST use `sns:`.

## 3. `ordns:` — reserved

`ordns:` is **reserved for a future standard** and is not defined here. Deployed resolvers already accept and strip the prefix on input; that acceptance remains (breaking inputs is never the answer), but applications MUST NOT emit `ordns:` URIs until the ORDNS standard defines them. Reservation now prevents squatting the prefix with incompatible meanings later.

## 4. `ord://` — the content scheme (browser)

The way a browser addresses **on-chain content**:

```
ord://<name>.<tld>          load the content coupled to this name
ord://<txid>[/<vout>]       load this inscription directly (vout defaults to 0)

ord://ordnet.web3
ord://dc54c20a…80ac/0
```

Rules:

1. **Name form.** The authority is resolved per ODNCA-STD-001 (signed answer, verification levels apply); the browser then loads the content currently coupled to the name — a 1Sat Ordinals inscription script in a sat, with the name↔content coupling read via the data-channel surface (ODNCA-STD-006). Content follows the coupling, so `ord://name.tld` stays valid across content updates.
2. **Txid form.** A 64-hex authority is an immutable direct address: it always loads that exact inscription, forever. Name form is *mutable reference*; txid form is *permanent reference*. Both are first-class.
3. **Sub-resources.** Paths after the authority (`ord://ordnet.web3/style.css`) address resources within the loaded content and are defined by the On-Chain Web standards (forthcoming); this document fixes only the scheme and authority.
4. **Status surfaces.** Content measures per ODNCA-POL-003 apply at load time: WARNING content sits behind the click-through, BLOCKED content renders the standard offline page — honestly labelled, never a fake "not found".

## 5. Disambiguation table

| INPUT | MEANING |
|---|---|
| `sns:alex.web3` | the name — action from context (pay, mail, open) |
| `sns:bob@alex.web3` | the mailbox — mail or mailbox payment |
| `ord://alex.web3` | the content coupled to the name, via resolution |
| `ord://<txid>` | that exact inscription, immutable |
| bare `alex.web3` in a wallet/browser field | treated as `sns:` / `ord://` respectively by field context |

## 6. Security considerations

- **Scheme spoofing.** Only these schemes are recognised; look-alikes (`snss:`, `0rd://`) MUST NOT be silently corrected — show them raw and unlinked.
- **Display.** Name display in any scheme follows the lookalike rules of ODNCA-STD-001 §10 (non-ASCII and mixed-script marking).
- **No credentials.** None of these schemes carries user-info beyond the mailbox; anything shaped like `user:pass@` MUST be rejected, not interpreted.