ODNCA-STD-003 · SNS-NAME-1

Name Format Standard for On-Chain Names

Status: v1.0 — Adopted

Applies to: the SNS namespace and the BSVmap namespace under the ODNCA root rules; the OpNS name shape is defined for interoperability.

Normative reference implementation: ORDnet indexer V31 (rules.toml + validation pipeline). This document describes the rules under which the existing corpus of ~600,000 names was indexed. It documents deployed reality; it does not introduce new rules and therefore requires no transition regime.

1. Scope

This standard defines what constitutes a **valid on-chain name**: the registration form, the normalization procedure, the structural and character rules, and the length bounds. Every conformant indexer and resolver MUST apply these rules identically; two parties reading the same chain under this standard MUST arrive at the same set of valid names.

Out of scope, by design:

- **Registrar sales policy.** Which names a registrar chooses to offer, restrict, or price is that registrar's own decision and no concern of this standard. A registrar MAY restrict its offering to any subset of valid names (for example, ASCII letters and digits only) and MAY change that policy at any time. Such restrictions never affect the validity of names registered directly on-chain.
- **Reserved names.** The chain recognises no reservations. A name is held by whoever validly claimed it first. Parties wishing to secure names do so the same way as everyone else: by claiming them.
- **Resolution.** How a valid name resolves to records, mailboxes, and payment destinations is defined in ODNCA-STD-001 (SNS Resolution Specification).

2. The two name shapes

The ODNCA root rules recognise two distinct name shapes, made deliberately unmistakable:

NAMESPACE	SHAPE	EXAMPLE
SNS	`name.tld` — exactly one dot, a name and a TLD	`alex.web3`
OpNS	bare name — no dot, no TLD	`alex`

An SNS name always contains exactly one dot. An OpNS name never contains a dot. No string can be valid in both namespaces; the presence or absence of the dot is the disambiguator. *(Normative rules in this document cover SNS and BSVmap, which the reference indexer processes today. The OpNS shape is defined here so that tooling can distinguish the namespaces; OpNS claim-validity rules follow the OpNS protocol's native genesis lineage.)*

The **BSVmap** namespace is a special numeric form within the dot-shaped family, defined in §7.

3. Registration form (SNS)

A valid SNS registration is an inscription whose body is a **JSON object** (JSON5 syntax accepted: comments and relaxed quoting are tolerated; the body must open with `{` after optional whitespace and comments) containing at minimum:

```
{ "p": "sns", "op": "reg", "name": "alex.web3" }
```

Rules:

1. `p` **MUST equal** `"sns"` and `op` **MUST equal** `"reg"`. Any other `op` value (e.g. `"transfer"`) is not a registration.
2. **The name comes exclusively from the** `name` **field.** A plain-text body that merely *looks* like a name — `test.com`, `index.html`, `style.css`, a filename inside a website inscription — is **not** a registration and never creates a claim. There is deliberately no plain-text registration form: this is what keeps 600k websites' worth of on-chain artefacts from polluting the namespace.
3. **Content-type is not determinative.** A valid registration counts whether it was inscribed as `application/json`, `text/plain`, `text/markdown`, or anything else. The gate is purely structural: valid UTF-8 → JSON(5) object → `p`/`op` match → valid `name`. Binary bodies (images, video) fail the UTF-8/object check naturally.
4. **The inscription output MUST carry exactly 1 satoshi** (the one-sat rule). The name is bound to that satoshi; ownership follows the sat (see ODNCA-STD-001).
5. **First seen wins.** Among valid registrations of the same canonical name, the earliest on-chain claim owns it. Later claims create no rights.

4. Normalization (canonicalization)

Before validation, the raw `name` value is normalized in three steps, in this order:

1. **Head:** cut the string at the first whitespace character (space, tab, newline); keep only what precedes it.
2. **Trim:** remove leading/trailing whitespace from the head.
3. **Lowercase:** apply Unicode lowercasing.

The result is the **canonical name** — the key under which the name is indexed, matched for first-seen-wins, and resolved. Consequences:

- `ALEX.web3` and `alex.web3` are the same name; the first valid claim of either owns `alex.web3`.
- `satoshi.web3 some trailing text` registers `satoshi.web3` (the trailing text is discarded, the registration is not rejected).
- `foo bar.web3` normalizes to `foo` — which then fails the structure rule (§5) and is invalid.

Indexers MUST additionally retain the **raw name** (the pre-lowercase head, original case preserved) alongside the canonical key. This is a security feature: it allows detection of visually deceptive registrations (hidden or look-alike characters) after the fact. See §8.

---

5. Structure rule (SNS)

The canonical name MUST match:

```
^[^\s.]+\.[^\s.]+$
```

In words: **one dot, exactly** — with a non-empty name part before it and a non-empty TLD part after it, and **no whitespace anywhere**. Zero dots is invalid (that shape belongs to OpNS). Two or more dots is invalid: there are no subdomains at the registration layer.

---

6. Character and length rules

SNS-NAME-1 is deliberately permissive at the character level and strict at the structural level:

1. **Allowed:** every UTF-8 character **except** whitespace, the dot (beyond the single separator), and control characters. This includes accented letters (`café.web3`), non-Latin scripts (`中文.web3`, `العربية.web3`), symbols (`a:b.web3`), and emoji (`🔥.web3`). All such names are valid and part of the indexed corpus's rule set since genesis.
2. **Rejected:** any canonical name containing a **control character** (e.g. NUL, 0x01) is invalid.
3. **Length:** the canonical name MUST be **1 to 2048 bytes** measured in UTF-8 octets (not characters). Over-length names are rejected, never truncated — truncation could collide a junk name into a different valid name.
4. **Case:** validity is judged, and uniqueness enforced, on the lowercase canonical form only.

---

7. The BSVmap namespace

BSVmap names are a numeric special form, inscribed as a plain-text body:

```
<number>.bsvmap        e.g.  900000.bsvmap
```

Rules: the body MUST match `^[0-9]+\.bsvmap$` (after lowercase-trim normalization); first seen wins; one-sat rule applies; and — the **minimum-height rule** — a claim on `<N>.bsvmap` is only valid if inscribed at block height ≥ N. You cannot claim a block that does not yet exist; first-seen-wins operates only among valid claims.

Because `.bsvmap` is a protocol of its own, the extension is **excluded from SNS**: a JSON SNS registration whose name ends in `.bsvmap` is invalid as an SNS name.

---

8. Security considerations

Confusables and spoofing. Permissive character rules mean visually similar names can be distinct claims: `apple.web3` (Latin) and `apple.web3` (Cyrillic a/p) are different canonical names, both valid. The standard's position:

- The **chain layer** does not and cannot adjudicate visual similarity; validity is mechanical.
- The **raw name retention** (§4) gives every indexer the evidence needed to flag mixed-script and confusable registrations.
- **Display layers** (wallets, portals, resolvers rendering names to humans) SHOULD visually distinguish names containing characters outside `a–z 0–9`, for example by highlighting, script labels, or punycode-style disclosure. This is guidance, not a validity rule.
- **Registrars** MAY restrict their sales offering to a conservative character subset. (The founding registrar currently offers letters-and-digits names only — a policy choice at that layer, changeable at that layer.)

Junk resistance. The JSON-only registration form (§3.2), the one-sat rule, the control-character rejection, and the reject-don't-truncate length rule together ensure that website artefacts, binary content, and malformed bodies can never become name claims or corrupt the index.

---

9. Test vectors

The normative machine vectors are the frozen indexer vector set (36 vectors, published with the conformance suite). The table below gives the human-readable summary; a conformant implementation MUST agree with every row.

#	INPUT (REGISTRATION `NAME` UNLESS NOTED)	CANONICAL RESULT	VALID?	RULE
1	`alex.web3`	`alex.web3`	✓	baseline
2	`ALEX.web3`	`alex.web3`	✓	lowercase (§4)
3	`satoshi.web3 extra text`	`satoshi.web3`	✓	head-strip (§4)
4	`café.web3`	`café.web3`	✓	permissive chars (§6.1)
5	`中文.web3`	`中文.web3`	✓	permissive chars (§6.1)
6	`💩.web3`	`💩.web3`	✓	permissive chars (§6.1)
7	`a:b.web3`	`a:b.web3`	✓	permissive chars (§6.1)
8	`noDotHere`	`nodothere`	✗	no dot → not SNS (§5)
9	`a.b.c`	`a.b.c`	✗	two dots (§5)
10	`foo bar.web3`	`foo`	✗	head-strip leaves no dot (§4, §5)
11	`` (empty)	``	✗	length < 1 byte (§6.3)
12	`ab␇cd.web3` (control char)	—	✗	control character (§6.2)
13	name of 2049+ bytes	—	✗	over 2048 bytes, rejected not truncated (§6.3)
14	`name.bsvmap` via SNS JSON	—	✗	extension excluded from SNS (§7)
15	plain-text body `test.com` (no JSON)	—	✗	not a registration (§3.2)
16	JSON with `op:"transfer"`	—	✗	not a registration (§3.1)
17	registration output ≠ 1 sat	—	✗	one-sat rule (§3.4)
18	plain text `900000.bsvmap` at height ≥ 900000	`900000.bsvmap`	✓	BSVmap (§7)
19	plain text `900000.bsvmap` at height < 900000	—	✗	minimum-height rule (§7)

10. Conformance

An implementation is **SNS-NAME-1 conformant** when it reproduces the exact valid/invalid verdict and canonical key for every vector in the frozen vector set. Conformance is technical and testable; no permission or committee is involved. Run the vectors, match the answers.
