Canonical Document Header (Informative / Non-Normative Metadata)

- Document ID: SC-HDR-000001
- Canonical Name: Structural Coherence Document Header Specification
- Document Class: SPEC
- Version: 0.4.0
- Issuance ID: v11
- Release Status: ISSUED
- Effective Date: 2026-03-19
- Last Updated: 2026-03-19
- Issuing Authority: Coherence Research
- DOI / Persistent ID:
- Supersedes: SC-HDR-000001 (Issuance ID: v10)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v11)
- Integrity:
  - SHA-256: 7ae85cc4eba408ba7602434be7daaacb24223a7075c97dba296d682f6c74f54d
  - Release Tag: 2026-03-19___v11

Document Class Extension: SPEC (Informative / Non-Normative Metadata)

SPEC::Header-Schema-Version: 0.4

SPEC::Standard: Structural Coherence

SPEC::Spec-Subclass: header-spec

SPEC::Revisability: PATCH | MINOR | MAJOR (Versioned Only)

SPEC::Canonical-Dependencies: SC-CORE-000001 (v39c), SC-AXIOM-000001 (v11k)

Structural Coherence Document Header Specification

1. Scope

This document defines the **normalized base header** required for Structural Coherence artifacts and the **document class extension header blocks** used to specialize the base header for different artifact types.

This header system is **method-intrinsic**: every canon artifact MUST include a canonical header so that identity, provenance, issuance posture, and integrity can be audited without reliance on filenames or external registries.

2. Definitions

2.1 Header Surface

The *header surface* of an artifact is the contiguous block of text beginning with the section heading:

`## Canonical Document Header (Informative / Non-Normative Metadata)`

and ending immediately before the next `---` delimiter.

2.2 Base Header

The *Base Header* is the required set of key/value fields inside the Canonical Document Header section.

2.3 Document Class Extension

A *Document Class Extension* is a second required header section:

`## Document Class Extension: <CLASS> (Informative / Non-Normative Metadata)`

containing class-scoped keys that extend (but do not override) the Base Header.

2.4 Canonical Source Artifact

The *canonical source artifact* is the `.md` file in the canonical repository. It is the authoritative artifact from which all derived formats are produced. The `Integrity::SHA-256` field in the Base Header governs the canonical source artifact only.

2.5 Derived Artifact

A *derived artifact* is any format (PDF, HTML, or other rendered output) produced from a canonical source artifact. Derived artifacts are not governed by the in-document `Integrity::SHA-256` field. Their integrity is governed by a Release Manifest (§3.2.5).

3. Base Header Requirements

3.1 Base Header Key Set (Required)

Every artifact MUST include exactly the following Base Header keys, in the order listed:

- Document ID
- Canonical Name
- Document Class
- Version
- Issuance ID
- Release Status
- Effective Date
- Last Updated
- Issuing Authority
- DOI / Persistent ID
- Supersedes
- Superseded By
- Header Schema
- Integrity

 CC BY-ND 4.0

Note: `Integrity` is a compound key with required subkeys `SHA-256` and `Release Tag` (see §3.2).

3.2 Formatting Rules

1) The Base Header MUST be represented as a Markdown list. For ordinary keys it MUST use `- Key: Value` lines.
2) The `Integrity` field is a required compound key and MUST be represented exactly as:

- `- Integrity:`
- followed by exactly two indented list items, in order:
  - `- SHA-256: <value>`
  - `- Release Tag: <value>`

3) Keys MUST match spelling and punctuation exactly.
4) Values MAY be empty in drafts, except:
   - Document ID
   - Document Class
   - Version
   - Issuance ID
   - Release Status
   - Effective Date
   - Issuing Authority
5) Issuance rule: if `Release Status` is `ISSUED`, then `Integrity::SHA-256` and `Integrity::Release Tag` MUST be non-empty.
6) Bundle rule: if an artifact is included in a distributed bundle accompanied by a manifest (e.g., `RELEASE_MANIFEST__<release>.json` or `KNOWLEDGE_MANIFEST__<release>.jsonl`), then `Integrity::SHA-256` MUST be populated and MUST match the manifest hash for that artifact.
7) Deterministic computation rule for `Integrity::SHA-256`: compute SHA-256 over the UTF-8 bytes of the complete artifact *as distributed* with the `Integrity` block present and the `- SHA-256:` value treated as empty (i.e., the line ends immediately after the colon). Newlines MUST be LF (`\n`). Validators MUST reproduce this by temporarily blanking the SHA-256 value before hashing. The `Integrity::SHA-256` field governs the canonical source artifact (the `.md` file) only. Derived artifacts (PDF, HTML, or other rendered formats) are not governed by this field; their integrity is governed by a Release Manifest (§3.2.5).

3.2.1 Canonicalization Requirements (Normative)

- Artifacts MUST be encoded as UTF-8.
- Newlines MUST be LF (`\n`).
- Artifacts MUST end with a final LF (`\n`).
- Tabs (`\t`) MUST NOT be used.
- Trailing whitespace at line ends MUST NOT be used.
- For hashing, the `- SHA-256:` line MUST contain no trailing characters after the colon when blanked (no spaces).

3.2.2 Release Status Enumeration (Normative)

`Release Status` MUST be one of:

- DRAFT
- ISSUED
- DEPRECATED
- SUPERSEDED
- WITHDRAWN

3.2.3 Issuance ID Format (Normative)

`Issuance ID` MUST match the following pattern:

- `v` followed by 2+ digits, optionally followed by a single lowercase letter (e.g., `v06`, `v11h`).

Validators SHOULD accept legacy issuance identifiers that do not match this pattern only in a documented compatibility mode.

3.2.4 Derived Artifact Integrity (Normative)

Derived artifacts (PDF, HTML, or other rendered formats produced from a canonical source) are distinct artifacts from the canonical source. They:

1. MUST NOT be expected to self-verify against the `Integrity::SHA-256` field in the document header, as that field governs the canonical source artifact only.
2. MAY embed the canonical source's `Document ID`, `Version`, `Issuance ID`, and `Integrity::SHA-256` in artifact metadata (e.g., PDF document properties) as a provenance reference. This constitutes identity attribution, not integrity verification of the derived artifact itself.
3. Their integrity MUST be governed by a Release Manifest (§3.2.5) when published as part of a release package.
4. A verifier checking a derived artifact (e.g., a downloaded PDF) MUST verify against the Release Manifest hash for that artifact, not the in-document `Integrity::SHA-256` field.

3.2.5 Release Manifest Requirement (Normative)

When one or more canonical artifacts are published as a release package, a Release Manifest MUST be produced. The Release Manifest:

1. MUST enumerate every artifact in the release package — both canonical source files and derived artifacts — by filename and SHA-256 hash.
2. MUST compute SHA-256 for canonical source files per §3.2 Rule 7 (blanked-hash source computation).
3. MUST compute SHA-256 for derived artifacts (PDF, HTML, etc.) over the full binary content of the derived file as distributed, with no blanking or transformation applied.
4. IS the authoritative integrity source for the release package. Verifiers checking a downloaded derived artifact MUST verify against the Release Manifest hash for that artifact.
5. MUST itself carry a canonical header and be published alongside the artifacts it governs.
6. SHALL follow the existing manifest format established in the repo (`RELEASE_MANIFEST__<release>.json`) extended to include both source and derived artifact entries with explicit `artifact-type` fields: `canonical-source` | `derived-pdf` | `derived-html`.

3.3 Uniqueness and Stability Rules

- Document ID MUST be globally unique across the canon.
- Document ID MUST NOT change after first issuance.

4. Document Class Requirements

4.1 Document Class Enumeration

`Document Class` MUST be one of:

- SPEC
- MODULE
- RCC
- LEDGER
- PAPER
- FRAMEWORK
- POLICY
- NOTE

4.2 Document Class Extension Block (Required)

Every artifact MUST include exactly one Document Class Extension block, immediately following the Base Header.

Extension keys MUST be namespaced as:

`<TOKEN>::<Key>: <Value>`

where `<TOKEN>` is the canonical namespace token for the artifact's Document Class, as defined in §5.

`<TOKEN>` SHALL be 2 to 4 uppercase ASCII alphanumeric characters.

5. Class Extension Specifications

5.0 Canonical Namespace Tokens

Each Document Class Extension block SHALL use the following canonical namespace token as the `<TOKEN>` prefix:

- SPEC → `SPEC`
- MODULE → `MOD`
- RCC → `RCC`
- LEDGER → `LEDG`
- PAPER → `PAPR`
- FRAMEWORK → `FWK`
- POLICY → `POL`
- NOTE → `NOTE`

5.1 SPEC Extension

Required keys:

- SPEC::Header-Schema-Version
- SPEC::Standard
- SPEC::Spec-Subclass
- SPEC::Revisability
- SPEC::Canonical-Dependencies

Optional keys:

- SPEC::Subtitle
- SPEC::Representational-Perspective
- SPEC::Discipline-Scope
- SPEC::Document-Posture
- SPEC::Patch-Label

5.2 MODULE Extension

Required keys:

- MOD::Header-Schema-Version
- MOD::Standard
- MOD::Module-ID (SHOULD equal Document ID)
- MOD::Canonical-Dependencies

Optional keys:

- MOD::Binding-Surface-Index
- MOD::Admitted-Vocabulary-Notes

5.3 RCC Extension

Required keys:

- RCC::Schema-Version
- RCC::Target-Document-ID
- RCC::Target-SHA-256
- RCC::Basis
- RCC::Result

Optional keys:

- RCC::Findings-Index
- RCC::Assumptions
- RCC::Tooling-Notes

5.4 LEDGER Extension

Required keys:

- LEDG::Schema-Version
- LEDG::Ledger-Type
- LEDG::Applies-To

Optional keys:

- LEDG::Resolution-Classes
- LEDG::Unresolved-Surface-Policy
- LEDG::Generation-Method

5.5 PAPER Extension

Required keys:

- PAPR::Schema-Version
- PAPR::Series
- PAPR::Dependency-Basis
- PAPR::Stability

Optional keys:

- PAPR::Abstract-Intent
- PAPR::Claims

5.6 FRAMEWORK Extension

Required keys:

- FWK::Schema-Version
- FWK::Basis
- FWK::Scope

Optional keys:

- FWK::Interfaces
- FWK::Invariants
- FWK::Compliance-Tests

5.7 NOTE Extension

Required keys:

- NOTE::Schema-Version
- NOTE::NonNormative-Only: TRUE

Optional keys:

- NOTE::Topic-Tags
- NOTE::Context

5.8 POLICY Extension

POLICY documents define stewardship constraints and distribution policies. They are informative / non-normative relative to SC-AS derivation, but they MAY be binding within the stewardship scope (e.g., trademark policy).

Required keys:

- POL::Schema-Version
- POL::NonDerivational-Only

- POL::Policy-Subtype
- POL::Applies-To

Optional keys:

- POL::Enforcement-Scope
- POL::Related-Artifacts

6. Conflict and Precedence Rules

1) The Base Header and Document Class Extension are the **canonical identity surface** for an artifact.
2) If any alternative metadata representation exists (e.g., YAML front matter), it MUST be treated as **derived**. In case of conflict, the Base Header/Extension govern.
3) Prohibited duplicate headers: canon artifacts MUST NOT include any additional header blocks that could be interpreted as authoritative metadata, including (but not limited to) `## Document Header` sections or YAML front matter. If such blocks are present, the artifact is non-conformant and MUST be corrected by removal (not interpretation).
4) Filenames are non-authoritative and MUST NOT be relied upon for canonical identity, status, or dependencies.
5) Filename convention: canonical artifacts SHALL use the filename pattern `{Document-ID}.md` (e.g., `SC-CORE-000001.md`). Issuance IDs, version numbers, and release tags SHALL NOT appear in filenames. Version identification is carried exclusively by the Base Header fields `Version` and `Issuance ID`. Reflexive Closure Certificates SHALL additionally include the target document's Issuance ID in the filename to preserve version-specific binding (e.g., `rcc__SC-CORE-000001__v39c__v01.md`).

7. Validation Rules (Minimum)

7.0 Parsing Discipline (Normative)

- Validators MUST treat fenced code blocks (Markdown triple-backtick regions) as opaque and MUST ignore header-like strings within them for the purposes of header uniqueness checks.
- Validators MUST treat only the first occurrence of `## Canonical Document Header (Informative / Non-Normative Metadata)` outside of fenced code blocks as the canonical header surface.

A header is conformant if:

- It contains the Base Header keys exactly once each (including the `Integrity` compound key with its two required subkeys).
- Document Class is one of the allowed enumerations.
- The Document Class Extension exists and uses only namespaced keys matching the Document Class.
- Required extension keys for that class are present.
- The artifact contains no prohibited duplicate header blocks (e.g., `## Document Header` sections or YAML front matter).
- If `Release Status` is `ISSUED`, the `Integrity::SHA-256` and `Integrity::Release Tag` values are non-empty.

7.1 Prohibited Canonical-Elevation Fields (Normative)

Artifacts governed by this specification SHALL NOT include header keys or metadata claims that assert canonical elevation, co-equal authority, or override relative to an anchor canonical set (e.g., keys or values indicating "Canonical: TRUE", "Elevated", "Equal Authority", "Overrides SC-CORE/SC-AXIOM"), except where such keys are explicitly defined by this specification. Validators SHALL treat the presence of such claims as nonconformant.

8. Examples

8.1 SPEC Example

```
## Canonical Document Header (Informative / Non-Normative Metadata)

- Document ID: SC-CORE-000001
- Canonical Name: The Coherence Core
- Document Class: SPEC
- Version: 0.2.1
- Issuance ID: v39c
- Release Status: ISSUED
- Effective Date: 2026-03-04
- Last Updated: 2026-03-04
- Issuing Authority: Coherence Research
- DOI / Persistent ID: 10.5281/zenodo.18039635
- Supersedes: SC-CORE-000001 (Issuance ID: v39b)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v11)
- Integrity:
  - SHA-256: 08566b068a5afd6ed61a6c0c2dc64ffe4895ea9e1e3e3e4a13d85709959c24f6
  - Release Tag: 2026-03-04__v39c

---

## Document Class Extension: SPEC (Informative / Non-Normative Metadata)

SPEC::Header-Schema-Version: 0.4

SPEC::Standard: Structural Coherence

SPEC::Spec-Subclass: core-spec

SPEC::Revisability: Versioned Only

SPEC::Canonical-Dependencies: SC-AXIOM-000001 (v11k)
```

8.2 RCC Example

```
## Canonical Document Header (Informative / Non-Normative Metadata)

- Document ID: RCC-SC-CORE-000001
```

```
- Canonical Name: Reflexive Closure Certificate - SC-CORE
- Document Class: RCC
- Version: 1.0.0
- Issuance ID: v01
- Release Status: ISSUED
- Effective Date: 2026-03-04
- Last Updated: 2026-03-04
- Issuing Authority: Coherence Research
- DOI / Persistent ID:
- Supersedes:
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v11)
- Integrity:
  - SHA-256: 08566b068a5afd6ed61a6c0c2dc64ffe4895ea9e1e3e3e4a13d85709959c24f6
  - Release Tag: 2026-03-04__v01

---

## Document Class Extension: RCC (Informative / Non-Normative Metadata)

RCC::Schema-Version: 1.0

RCC::Target-Document-ID: SC-CORE-000001 (v39c)

RCC::Target-SHA-256: [hash of target document]

RCC::Basis: SC-AS (SC-HDR-000001 v11; SC-SCOPE-000001 v04; SC-CORE-000001 v39c; SC-AXIOM-000001 v11k

RCC::Result: PASS
```

End of Document