Canonical Document Header (Informative / Non-Normative Metadata)

- Document ID: SC-HDR-000001
- Canonical Name: Structural Coherence Document Header Specification
- Document Class: SPEC
- Version: 0.5.0
- Issuance ID: v13
- Release Status: ISSUED
- Effective Date: 2026-03-29
- Last Updated: 2026-03-29
- Issuing Authority: Coherence Research
- Author: Jason Carroll, Coherence Research
- Abstract: Defines the normalized base header and document class extension blocks required for all Structural Coherence canonical artifacts, governing identity, provenance, issuance posture, and integrity.
- Language: eng
- License: CC-BY-4.0
- Change-Summary: Promoted Author, Abstract, Language, License, Change-Summary, Deprecation-Reason, Title, and Keywords to optional base header fields. Removed Author from class extension blocks. Added conditional normative rules for DEPRECATED and WITHDRAWN status. Extended validation rules with conditional checks. Added cross-class dependency field mapping. Updated canonical identity surface language in §6. Schema-breaking: existing documents with Author in extension blocks remain conformant but Author in base header takes precedence.
- DOI / Persistent ID:
- Supersedes: SC-HDR-000001 (Issuance ID: v12)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v13)
- Integrity:
  - SHA-256: 8b7b3788eeea853dddbc734ecc5bfb2cf319d9d84b9d20deb1737241dafc9cd1
  - Release Tag: 2026-03-29__v13

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

SPEC::Header-Schema-Version: 0.5

SPEC::Standard: Structural Coherence

SPEC::Spec-Subclass: header-spec

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

SPEC::Canonical-Dependencies: SC-CORE-000001 (v40), SC-AXIOM-000001 (v12)

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 canonical 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 and optional set of key/value fields inside the Canonical Document Header section. Required fields MUST be present in every artifact. Optional fields MAY be omitted; omission is not an error.

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).

2.6 Minimum Identity Surface

The *minimum identity surface* is the subset of base header fields sufficient to establish unique, unambiguous identity for an artifact without reference to any external registry or class extension: Document ID, Canonical Name, Document Class, Version, Issuance ID, Release Status, Effective Date, Issuing Authority, and Integrity.

2.7 Extended Identity Surface

The *extended identity surface* is the full base header — minimum identity surface plus all populated optional fields — together with the Document Class Extension block. The extended identity surface is the canonical identity surface for the artifact as stated in §6.

3. Base Header Requirements

3.0 Schema Version Compatibility (Normative)

A change to SC-HDR is schema-breaking if it could cause a document conformant under a prior issuance of the same schema version to fail validation. Schema-breaking changes MUST increment the `Version` field (e.g., `0.4.0` → `0.5.0`).

A change to SC-HDR is non-breaking if all documents conformant under a prior issuance of the same schema version remain conformant after the change. Non-breaking changes — including new optional fields, clarifying text, and additive extension keys — MAY be issued under the same schema version with a new `Issuance ID` only.

Documents conformant under any prior issuance of schema version X remain conformant under all subsequent non-breaking issuances of schema version X. No reissuance of prior conformant documents is required for non-breaking issuances.

Version 0.5.0 schema-breaking change: The `Author` field is now a base header field. Documents that carry `Author` only in a class extension block remain conformant but are considered to have an incomplete base header. When both a base header `Author` field and a class extension `Author` field are present, the base header `Author` field governs and the extension `Author` field is treated as a deprecated alias.

3.1 Base Header Key Set

Every artifact MUST include the required base header keys listed below, in the order listed. Optional base header keys MAY appear after `Issuing Authority` and before `DOI / Persistent ID`, in the order listed. Optional keys that are omitted MUST NOT leave blank lines in their place — omission means the key is absent entirely.

Required keys (MUST be present in every artifact):

- Document ID
- Canonical Name
- Document Class
- Version
- Issuance ID
- Release Status

- Effective Date
- Last Updated
- Issuing Authority

Optional keys (MAY be present; listed in canonical order):

- Author
- Title
- Abstract
- Keywords
- Language
- License
- Change-Summary
- Deprecation-Reason

Required keys (continued, MUST be present in every artifact):

- DOI / Persistent ID
- Supersedes
- Superseded By
- Header Schema
- Integrity

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, 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.

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

Conditional normative rules by status:

- If `Release Status` is `ISSUED`: `Integrity::SHA-256` and `Integrity::Release Tag` MUST be non-empty; `Effective Date` MUST be non-empty.
- If `Release Status` is `SUPERSEDED`: `Superseded By` SHOULD be non-empty, identifying the superseding document.
- If `Release Status` is `DEPRECATED`: `Deprecation-Reason` SHOULD be non-empty, stating why the document was deprecated and what, if anything, should be used instead.
- If `Release Status` is `WITHDRAWN`: `Deprecation-Reason` SHOULD be non-empty, stating why the document was 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 Optional Field Semantics

Author

Records the individual author(s) of the artifact for citation and attribution purposes.

Format: `Firstname Lastname` or `Firstname Lastname [ORCID], Affiliation`. Multiple authors separated by semicolons. ORCID format: 19-character string `XXXX-XXXX-XXXX-XXXX`.

Example: `Jason Carroll [0000-0000-0000-0000], Coherence Research`

When `Author` is present in both the base header and a class extension block, the base header governs.

Title

The formatted publication title for the artifact, suitable for use in external registries and citations (e.g., Zenodo title field). Distinct from `Canonical Name`, which is the raw internal identifier. When present, the pipeline uses `Title` for external publication. When absent, the pipeline SHOULD generate a formatted fallback from `Canonical Name` in the form: `{Standard} — {Canonical Name title-cased} ({Document ID})`.

Example: `Structural Coherence — The Coherence Core Specification (SC-CORE-000001)`

Abstract

A brief description of the artifact's scope and purpose, suitable for use in external publication metadata (e.g., Zenodo description field). Plain text. One to three sentences recommended.

Keywords

A comma-separated list of subject tags for discoverability in external registries and indexes. Should include the document class, standard name, document ID, and relevant subject matter terms.

Example: `structural coherence, axiomatic system, open standard, SC-CORE-000001, spec`

Language

The primary language of the artifact's content. Use ISO 639-3 language codes (e.g., `eng` for English, `fra` for French). Defaults to `eng` if omitted.

License

The license governing distribution and reuse of the artifact. Use SPDX license identifiers (e.g., `CC-BY-4.0`, `MIT`, `Apache-2.0`). Coherence Research artifacts default to `CC-BY-4.0` if omitted.

Change-Summary

A brief human-readable summary of what changed from the version identified in `Supersedes`. Present when the artifact supersedes a prior issuance. Omit on first issuance.

Deprecation-Reason

States why the artifact was deprecated or withdrawn, and what alternative (if any) should be used. Present when `Release Status` is `DEPRECATED` or `WITHDRAWN`. SHOULD be non-empty in those cases per §3.2.2.

3.4 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

5.0.1 Cross-Class Dependency Field Mapping

Different document classes use different extension keys to express dependency relationships. The following table maps the class-specific dependency field to its base-header equivalent for tools that must traverse dependencies across classes without class-specific knowledge:

Document Class	Extension dependency field	Semantics
SPEC	SPEC::Canonical-Dependencies	Canonical specs this spec depends on
FRAMEWORK	FWK::Basis	SC-AS specs and versions this framework is grounded in
PAPER	PAPR::Dependency-Basis	SC-AS specs underpinning the paper's claims
MODULE	MOD::Canonical-Dependencies	Canonical specs this module depends on
RCC	RCC::Basis	Specs and versions the certificate covers
LEDGER	LEDG::Applies-To	The canonical material the ledger governs
POLICY	POL::Applies-To	The scope to which the policy applies
NOTE	(none — notes carry no dependencies)	N/A

A dependency graph tool MUST use this mapping to locate dependency fields per class. No cross-class normalization of dependency field names is required; this table is the normative mapping.

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

Note: `Author` was previously listed as an optional SPEC extension key. It is now a base header optional field. Existing documents with `SPEC::Author` remain conformant; the base header `Author` field takes precedence when both are present.

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

Note: `Author` was previously listed as an optional MOD extension key. It is now a base header optional field.

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

Note: `Author` was previously listed as an optional PAPR extension key. It is now a base header optional field. `PAPR::Abstract-Intent` remains as a class-specific optional field for papers that require more detailed intent framing beyond the base `Abstract` field.

5.6 FRAMEWORK Extension

Required keys:

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

Optional keys:

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

Note: `Author` was previously listed as an optional FWK extension key. It is now a base header optional field. `FWK::Scope` remains as a class-specific required field for frameworks; it carries richer scope semantics than the base `Abstract` field and both may be present simultaneously.

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 together constitute the extended identity surface (§2.7) for an artifact. The minimum identity surface (§2.6) is the subset of the base header sufficient to establish unique identity without class-specific extension fields.
2) The minimum identity surface MUST be sufficient to uniquely identify an artifact — its Document ID, class, version, issuance, status, date, authority, and integrity — without reading the extension block. Optional base header fields (Author, Abstract, Language, License, Change-Summary, Deprecation-Reason) are part of the extended identity surface and enrich identity for external publication and tooling purposes.
3) If any alternative metadata representation exists (e.g., YAML front matter), it MUST be treated as derived. In case of conflict, the Base Header and Extension govern.
4) 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).
5) Filenames are non-authoritative and MUST NOT be relied upon for canonical identity, status, or dependencies.
6) 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-00000`

7) When `Author` is present in both the base header and a class extension block, the base header `Author` field is authoritative. The extension-block `Author` field is treated as a deprecated alias and SHOULD be removed on next issuance.

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.

7.1 Unconditional Checks

A header is conformant if:

- It contains all required Base Header keys exactly once each (including the `Integrity` compound key with its two required subkeys).
- Document Class is one of the allowed enumerations (§4.1).
- 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).
- Issuance ID matches the format defined in §3.2.3.
- Optional base header keys, when present, appear in the canonical order defined in §3.1.

7.2 Conditional Checks (Normative)

The following checks apply conditionally based on field values:

Condition	Required check
`Release Status` = `ISSUED`	`Integrity::SHA-256` MUST be non-empty
`Release Status` = `ISSUED`	`Integrity::Release Tag` MUST be non-empty
`Release Status` = `ISSUED`	`Effective Date` MUST be non-empty
`Release Status` = `SUPERSEDED`	`Superseded By` SHOULD be non-empty
`Release Status` = `DEPRECATED`	`Deprecation-Reason` SHOULD be non-empty
`Release Status` = `WITHDRAWN`	`Deprecation-Reason` SHOULD be non-empty
`Supersedes` is non-empty	`Change-Summary` SHOULD be non-empty
`Author` present in base header AND in class extension	Base header governs; validator SHOULD warn of deprecated alias
`Language` is non-empty	Value SHOULD be a valid ISO 639-3 language code
`License` is non-empty	Value SHOULD be a valid SPDX license identifier
`Title` is non-empty	Used by pipeline for external publication; takes precedence over generated fallback
`Keywords` is non-empty	Value SHOULD be a comma-separated list of subject tags
Base header `Author` is empty	Validator SHOULD warn if a class extension `Author` field is present (migration opportunity)

7.3 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 (full optional fields)

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

- Document ID: SC-CORE-000001
- Canonical Name: THE COHERENCE CORE SPECIFICATION
- Document Class: SPEC
- Version: 1.2.5
- Issuance ID: v40
- Release Status: ISSUED
- Effective Date: 2026-03-04
- Last Updated: 2026-03-29
- Issuing Authority: Coherence Research
- Author: Jason Carroll [0000-0000-0000-0000], Coherence Research
- Title: Structural Coherence — The Coherence Core Specification (SC-CORE-000001)
- Abstract: Defines the axiomatic primitives, foundational conditions, valid
 ↪  emergence conditions, and admissibility rules constituting the minimal closed
 ↪  kernel of Structural Coherence — a universal structural invariant governing
 ↪  admissible configurations at any scale.
- Keywords: structural coherence, axiomatic system, open standard, admissibility,
 ↪  SC-CORE-000001, spec
- Language: eng
- License: CC-BY-4.0
- Change-Summary: Corrected PDF rendering — verbatim block overflow on p24-26 and
 ↪  display math overflow on p12 fixed. Source normalized to SC-AUTH v2.0 markdown
 ↪  grammar. No content changes.
- DOI / Persistent ID: 10.5281/zenodo.18039635
- Supersedes: SC-CORE-000001 (Issuance ID: v39d)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v13)
- Integrity:
  - SHA-256: db8ac460bae16dd5b146d3d4153d851a3a261fb7b27e79556cf83c022f14cf32
  - Release Tag: 2026-03-29__v40

---

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

SPEC::Header-Schema-Version: 0.5

SPEC::Standard: Structural Coherence

SPEC::Spec-Subclass: core-spec
```

```
SPEC::Revisability: Versioned Only

SPEC::Canonical-Dependencies: SC-AXIOM-000001 (v12)
```

8.2 SPEC Example (minimal — required fields only)

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

- Document ID: SC-CORE-000001
- Canonical Name: THE COHERENCE CORE SPECIFICATION
- Document Class: SPEC
- Version: 1.2.5
- Issuance ID: v40
- Release Status: ISSUED
- Effective Date: 2026-03-04
- Last Updated: 2026-03-29
- Issuing Authority: Coherence Research
- DOI / Persistent ID: 10.5281/zenodo.18039635
- Supersedes: SC-CORE-000001 (Issuance ID: v39d)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v13)
- Integrity:
  - SHA-256: db8ac460bae16dd5b146d3d4153d851a3a261fb7b27e79556cf83c022f14cf32
  - Release Tag: 2026-03-29__v40

---

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

SPEC::Header-Schema-Version: 0.5

SPEC::Standard: Structural Coherence

SPEC::Spec-Subclass: core-spec

SPEC::Revisability: Versioned Only

SPEC::Canonical-Dependencies: SC-AXIOM-000001 (v12)
```

8.3 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
- Author: Jason Carroll [0000-0000-0000-0000], Coherence Research
- DOI / Persistent ID:
- Supersedes:
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v13)
- Integrity:
  - SHA-256:
  - 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 (v40)

RCC::Target-SHA-256: db8ac460bae16dd5b146d3d4153d851a3a261fb7b27e79556cf83c022f14cf32

RCC::Basis: SC-AS (SC-HDR-000001 v13; SC-SCOPE-000001 v04; SC-CORE-000001 v40;
 ↪  SC-AXIOM-000001 v12)

RCC::Result: PASS
```

8.4 FRAMEWORK Example

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

- Document ID: SC-FCALC-000001
- Canonical Name: Structural Coherence Formal Operational Calculus -- Formal Theory
- Document Class: FRAMEWORK
- Version: 0.7.0
- Issuance ID: v04
- Release Status: ISSUED
- Effective Date: 2026-03-27
- Last Updated: 2026-03-27
- Issuing Authority: Coherence Research
- Author: Jason Carroll, Coherence Research
- Abstract: Formal operational theory over SC-AS primitives providing a calculus for
 ↪  structural verification and admissibility checking.
- Language: eng
- License: CC-BY-4.0
- Change-Summary: Extended σ-step pipeline with Γ_fb convergence loop. Added V1-V10
 ↪  verification conditions.
- DOI / Persistent ID: 10.5281/zenodo.XXXXXXXX
- Supersedes: SC-FCALC-000001 (Issuance ID: v03)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v13)
- Integrity:
  - SHA-256:
  - Release Tag: 2026-03-27__v04
```

```
---

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

FWK::Schema-Version: 1.0

FWK::Basis: SC-AS (SC-CORE-000001 v40; SC-AXIOM-000001 v12; SC-HDR-000001 v13;
 ↪  SC-SCOPE-000001 v04)

FWK::Scope: Formal operational theory over SC-AS primitives
```

8.5 DEPRECATED Document Example

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

- Document ID: SC-PCF-000001
- Canonical Name: PCF METHODOLOGY SPECIFICATION
- Document Class: SPEC
- Version: 0.5.0
- Issuance ID: v06
- Release Status: DEPRECATED
- Effective Date: 2026-01-15
- Last Updated: 2026-03-29
- Issuing Authority: Coherence Research
- Deprecation-Reason: Superseded by the PCF Skill Suite (pcf-decomposition,
 ↪  pcf-alignment, pcf-designer, pcf-analysis, pcf-sme). The skill suite replaces
 ↪  this document as the operational delivery mechanism for PCF methodology.
- DOI / Persistent ID:
- Supersedes: SC-PCF-000001 (Issuance ID: v05)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v13)
- Integrity:
  - SHA-256:
  - Release Tag: 2026-01-15__v06

---

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

SPEC::Header-Schema-Version: 0.5

SPEC::Standard: Structural Coherence

SPEC::Spec-Subclass: methodology-spec

SPEC::Revisability: Versioned Only

SPEC::Canonical-Dependencies: SC-CORE-000001 (v40)
```

End of Document