Canonical Document Header (Informative / Non-Normative Metadata)

- Document ID: SC-DOC-000001
- Canonical Name: Structural Coherence Document Authoring Specification
- Document Class: SPEC
- Version: 1.0.0
- Issuance ID: v02
- Release Status: ISSUED
- Effective Date:
- Last Updated: 2026-04-01
- Issuing Authority: Coherence Research
- Author: Jason Carroll, Coherence Research
- Title: Structural Coherence – Document Authoring Specification (SC-DOC-000001)
- Abstract: Defines the complete authoring surface for all Structural Coherence documents. Governs expression principles, content types, block and inline primitives, semantic containers, composition and nesting rules, format compatibility, class-specific content requirements, section templates, content ordering, and pipeline interface. Supersedes SC-AUTH-000001.
- Keywords: structural coherence, document authoring, formatting, grammar, pipeline, SC-DOC-000001, spec
- Language: eng
- License: CC-BY-4.0
- DOI / Persistent ID: 10.5281/zenodo.19375944
- Supersedes: SC-DOC-000001 (Issuance ID: v01)
- Superseded By:
- Header Schema: SC-HDR-000001 (Issuance ID: v13)
- Integrity:
  - SHA-256: eaa375511104324c546aceec4d862781c97f1bbbc95501d708ba7e3a5925e1dd
  - Release Tag: 2026-04-01__v02

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

SPEC::Header-Schema-Version: 0.5

SPEC::Standard: Structural Coherence

SPEC::Spec-Subclass: authoring-spec

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

SPEC::Canonical-Dependencies: SC-CORE-000001 (v40), SC-AXIOM-000001 (v12), SC-HDR-000001 (v13)

Structural Coherence Document Authoring Specification

0. Scope and Invariants

0.1 Scope

This specification defines the complete authoring surface for documents in the Structural Coherence ecosystem. It governs every aspect of document composition that an author controls: what content types exist, how they are expressed, how they compose, which combinations are safe, and what each document class requires.

The governing principle: one content type, one expression. If the grammar is followed, the pipeline is deterministic. Violations of the grammar are the single root cause of pipeline failures. Patterns not declared in this specification are not permitted. Patterns that produce pipeline failures despite conformance to this specification indicate a specification gap requiring grammar extension.

This specification is the author-facing contract. It declares what authors must know and follow. It does not declare how the pipeline renders – that is the concern of SC-PIPE (the rendering contract). The interface between this specification and SC-PIPE is declared in Section 11.

0.2 Invariants

This specification commits to the following:

1) Completeness. Every content pattern that may appear in an SC-AS document is declared in this specification. If a pattern is not declared here, it is not permitted.
2) Determinism. Following this grammar produces identical rendering output regardless of document content, length, or complexity. No per-document intervention is required.
3) Composability. The composition and compatibility rules (Sections 6 and 7) are exhaustive. Every combination of block and inline types has a declared status: SAFE, CAUTION, or PROHIBITED.
4) Class completeness. Every document class has declared content requirements (Section 8) and a section template (Section 9). An author can produce a conformant document of any class by following the template and satisfying the content requirements.

0.3 Relationship to Other Specifications

- SC-HDR-000001 governs document identity (headers, metadata, integrity). This specification does not duplicate header rules – it references SC-HDR for all header-related requirements.
- SC-AUTH-000001 is superseded by this specification. All content from SC-AUTH v2.1 is incorporated here with extensions.
- SC-PIPE (future) will declare the rendering contract. Section 11 of this specification declares the author-facing pipeline interface until SC-PIPE is issued.

0.4 Enforcement

- preflight.sh enforces this grammar mechanically at the pipeline gate. No document renders unless it passes preflight.
- export.sh gates on preflight before invoking Pandoc.

To run preflight independently:

```
bash release/export/preflight.sh path/to/document.md
```

1. Definitions

Block type. A content element that occupies one or more complete lines and is separated from adjacent blocks by blank lines. Block types MUST NOT nest inside inline contexts.

Inline type. A content element embedded within a block. Inline types MUST NOT stand alone – they MUST always appear inside a block context.

Semantic container. A multi-block pattern with a declared structure: a label line, body content, and optionally an end marker. Semantic containers use existing block types (prose, display math, lists) but the pattern itself is declared so that authors know it and preflight can validate it.

Content type. The union of block types, inline types, and semantic containers. Every line in a source document is part of exactly one content type.

Expression. The single correct markdown representation for a content type. Each content type has exactly one expression. Alternative representations are not permitted.

Format primitive. A block type or inline type considered as a formatting element rather than a semantic element. Sections 6 and 7 use this term when discussing how formatting elements compose and interact.

Containment. The relationship where one content type appears inside another. Section 6 declares which containment relationships are valid.

Compatibility. The interaction status when two format primitives co-occur. Section 7 declares the status (SAFE, CAUTION, PROHIBITED) for every pairwise combination.

Document class. The enumerated type of a document (SPEC, MODULE, RCC, LEDGER, PAPER, FRAMEWORK, POLICY, NOTE) as defined by SC-HDR-000001. Each class has specific content requirements (Section 8) and a section template (Section 9).

Pipeline stress. The degree to which a document class exercises the format compatibility surface. High-stress classes (SPEC, FRAMEWORK, PAPER) use math-heavy content that exercises most compatibility cells. Low-stress classes (GUIDE, POLICY, NOTE, RCC) use prose-heavy content with fewer format interactions.

2. Expression Principles

These principles govern how authors express content. They are not formatting rules – they are structural commitments that determine whether a document is coherent.

2.1 Constructive Mode

Every claim in a specification MUST be constructed from declared primitives. Authors MUST state what IS, not what is NOT. Negative scope declarations (Section 5.6) are the sole exception – they establish boundaries by stating what the document does not address.

When introducing a new construct, the author MUST show how it is built from existing elements. If the construct cannot be built from existing elements, it requires admission through the austerity gate (2.2).

2.2 Austerity Discipline

Every term, construct, and commitment in a document MUST be necessary, representable, and enforceable within the SC-AS framework. This applies to the author's own writing:

- Necessary. Does this construct need to exist? Can the document's claims be made without it? If it can be eliminated without loss, it MUST be eliminated.
- Representable. Can this construct be expressed using SC-AS primitives? If it requires importing external frameworks, methods, or mathematical machinery as primitives, it is inadmissible.
- Enforceable. Can conformance to this construct be verified? If a rule cannot be checked (mechanically or by review), it is not a rule – it is aspiration.

2.3 Concept Introduction via PCF

When introducing a new concept, the author MUST follow the structural order: state its presence, then its connections, then its flow.

1) Presence. What is it? Name it. Define it. State what it IS.
2) Connection. What does it relate to? State dependencies, interfaces, and structural relationships to existing constructs.
3) Flow. How does it operate? State dynamics, transformations, and processes.

This ordering applies to individual definitions, to sections, and to the document as a whole. A reader encountering a concept SHOULD know what it is before learning what it connects to, and what it connects to before learning how it behaves.

2.4 Register Conventions

SC-AS documents use RFC 2119 keywords with structural grounding:

- MUST / SHALL – structural necessity. The system cannot be coherent without this. Violation produces a conformance failure.
- SHOULD – structural recommendation. The system is coherent without this, but degraded. Violation produces a warning.
- MAY – structural permission. The system is coherent either way. Presence or absence is a design choice.

All normative content MUST use present tense and third person. Imperative mode is acceptable for procedural instructions (checklists, commands). First person MUST NOT be used in specifications.

Every normative constraint in this specification and in documents governed by it MUST use one of the keywords above. Unqualified imperatives (e.g., bare "do not" without MUST NOT, or bare "always" without MUST) carry no normative weight and MUST NOT appear in normative content.

3. Block Types

The following eight block types constitute the core grammar. Every line in a source document is part of one of these block types or a semantic container (Section 5) built from them.

3.1 Prose Paragraph

What it is: Continuous natural language text. The default block type.

Expression: Plain text, flush left, separated from adjacent blocks by a blank line. No leading whitespace.

```
For all admissible configurations X, structural differentials are
regulated such that all relational differentials remain contained.
```

Pandoc renders prose paragraphs as standard paragraphs. Line breaks within the source are ignored – Pandoc reflows to the page width. Authors MUST NOT use line breaks to control visual width in the source.

3.2 Heading

What it is: A section title at a defined hierarchy level.

Expression: One to five # characters followed by a space and the heading text. No trailing #.

```
# Document Title         (H1 -- appears once, after extension block)
## Major Section         (H2)
### Subsection           (H3)
#### Minor subsection    (H4)
##### Term heading       (H5)
```

H1 MUST appear exactly once per document, immediately after the Document Class Extension block. H2 through H5 are used for structural hierarchy. Authors MUST NOT skip heading levels (e.g., H2 to H4 without an intervening H3).

3.3 Unordered List Item

What it is: A discrete item in a non-sequential list.

Expression: A hyphen followed by a space, then the item content. Nested items indent by 3 spaces per level.

```
- First item
- Second item
   - Nested under second
   - Also nested
      - Double nested
```

3-space nesting is the only correct nesting increment. Authors MUST NOT use 4-space indent for list nesting – CommonMark treats 4-space-indented lines as verbatim code blocks, which overflows the PDF right margin.

3.4 Ordered List Item

What it is: A discrete item in a sequential or enumerated list.

Expression: A number or letter followed by `)` and a space, then the item content. Nested items indent by 3 spaces per level.

```
1) First item
2) Second item
   (a) Sub-item
   (b) Sub-item
3) Third item
```

Primary enumeration MUST use `1)` style (number + parenthesis). Lettered sub-items MUST use `(a)` style. 3-space nesting applies identically to ordered lists.

3.5 Display Math

What it is: A mathematical expression presented on its own line, rendered in display mode.

Expression: `\[...\]` delimiters, with the opening `\[` and closing `\]` each on their own line, surrounded by blank lines. Math content lines are indented by 3 spaces.

```
\[
   X \preceq Y :\Longleftrightarrow \exists\, e:|X| \to |Y|
\]
```

Alternatively, `$...$` MAY be used for single-line display math:

```
$X \preceq Y$
```

Both forms require blank lines above and below. The 3-space indent on content lines inside `\[...\]` is correct – it is prose indentation within a display math context, not a list or verbatim trigger.

3.6 Code Block

What it is: Literal text rendered verbatim in a monospace environment, typically for examples, commands, or structured non-mathematical notation.

Expression: Triple backtick fences with a language identifier.

````
```bash
bash release/export/export.sh specs/canonical/SC-CORE-000001.md
```
````

````
```text
RCC-ID: <unique identifier>
Applies-To: SC-CORE-000001
```
````

Code blocks MUST include a language identifier. Authors SHOULD use `text` for plain structured content with no syntax highlighting. Language-specific identifiers (`bash`, `python`, `markdown`, etc.) SHOULD be used for language-specific content.

3.7 Table

What it is: Tabular data with column headers and rows.

Expression: Standard GFM pipe table syntax.

```
| Column A | Column B | Column C |
|----------|----------|----------|
| Value    | Value    | Value    |
| Value    | Value    | Value    |
```

The separator row MUST use hyphens only (no colons for alignment unless alignment is semantically meaningful). Tables MUST be used for reference data – variable maps, feature comparisons, structured inventories. Tables MUST NOT be used for content that is better expressed as prose.

Column limits: prose tables SHOULD NOT exceed 4 columns. Data tables with 5 or more columns are acceptable but expect reduced font size in PDF. Cell content SHOULD NOT exceed approximately 40 characters per cell in 4-column tables.

3.8 Horizontal Rule

What it is: A visual section break.

Expression: Exactly `---` on its own line, with a blank line above and below.

```

---

```

No other form (`***`, `___`) is permitted. `---` without surrounding blank lines MAY be misread by Pandoc as a YAML delimiter or table separator – the blank lines are part of the correct expression, not optional.

4. Inline Types

The following four inline types MAY appear within any block. They MUST NOT stand alone – they MUST always be embedded in a block context.

4.1 Inline Math

What it is: A mathematical expression embedded within prose.

Expression: `\(...\)` delimiters around the LaTeX expression.

```
Expressive capacity \(\mathcal{E}(X)\) is the ordering class of X.
```


All LaTeX math commands (`\mathcal`, `\mathrm`, `\preceq`, `\rightarrow`, etc.) MUST appear inside `\(...\)` delimiters. Bare LaTeX in prose – without delimiters – is not recognized by Pandoc as math and renders as literal backslash-prefixed text.

Unicode math symbols MAY appear in prose without delimiters when used as isolated symbols within natural language. The export template maps these to their LaTeX equivalents. For the complete supported symbol list, see Appendix A.

Critical distinction – symbol vs. expression:

A single Unicode math symbol embedded in a sentence is a *symbol in prose*. Two or more Unicode math symbols forming a logical or mathematical statement constitute an *expression* and MUST be delimited.

Correct – isolated symbol in prose:

```
If X is in S then the configuration is admissible.
```

Correct – expression uses inline math delimiters:

```
If \(Drivers_\rho(\Psi) \neq \emptyset\) then further reduction exists.
```

Wrong – expression written as bare Unicode prose (creates unbreakable runs in PDF):

```
D(A) \precsim_D D(B) \iff Out(A) \neq \emptyset \land Out(B) \neq \emptyset
```

Promotion rule: Any expression containing a relational operator combined with variables or function notation MUST use `\(...\)` inline math delimiters. Expressions with three or more operators, or expressions longer than approximately 60 characters, SHOULD be promoted to display math (`\[...\]`).

4.2 Inline Code

What it is: A literal identifier, path, command, or value embedded in prose.

Expression: Single backtick delimiters.

```
The field `doc_id` is required. Run `restamp-sha.sh` to update.
```

Inline code is reserved for field names, file paths, commands, language tokens, and exact string values. Inline code MUST NOT be used for emphasis or decoration.

Promotion rule: Inline code spans exceeding approximately 60 characters SHOULD be promoted to code blocks. File paths and commands with arguments SHOULD use code blocks rather than inline code.

4.3 Bold

What it is: Strong structural emphasis on a term or phrase.

Expression: `**double asterisks**` around the text.

```
The governing principle: **one content type, one expression.**
```

Bold SHOULD be reserved for key terms being defined, critical constraints, and structural pivots. Bold MUST NOT be used for general emphasis or decoration. Math symbols and expressions MUST NOT appear inside bold markup – math content MUST remain outside emphasis (see Section 7).

4.4 Italic

What it is: Soft emphasis, typically for a term being introduced or referenced.

Expression: `*single asterisks*` around the text.

```
This is called *canonical conformance* throughout the specification.
```

Italic SHOULD be used for terms at their point of introduction and for titles of referenced documents. Italic and bold MUST NOT be used interchangeably.

5. Semantic Containers

Semantic containers are multi-block patterns that use existing block types but follow a declared structure. They are not new block types – they are recognized patterns with validation requirements.

5.1 Definition Block

What it is: A formal definition of a term or construct.

Structure:

```
**Definition X.Y.Z (Name).** Statement of the definition in prose,
which may span multiple lines.

**SC-AS grounding:** Trace to SC-AS primitives establishing that
this definition is admissible.
```

The label line uses bold with the pattern `**Definition {number} ({name}).**` followed by the definition statement. The SC-AS grounding block is REQUIRED for SPEC and FRAMEWORK documents. It MAY be omitted for other document classes.

Definition numbering MUST follow section numbering: Definition 3.1 is the first definition in Section 3, Definition 3.2 is the second, and so on.

5.2 Theorem Block

What it is: A formal claim that is subsequently proved.

Structure:

```
**Theorem X.Y.Z (Name).** Statement of the theorem.
```

The label line uses bold with the pattern `**Theorem {number} ({name}).**` followed by the theorem statement. Every theorem MUST have an accompanying proof (Section 5.4).

5.3 Lemma and Corollary Blocks

What it is: Supporting claims (lemmas precede the theorem they support; corollaries follow).

Structure: Identical to Theorem (5.2) with the label `**Lemma {number} ({name}).**` or `**Corollary {number} ({name}).**` respectively. Lemmas MUST have proofs. Corollaries SHOULD have proofs or explicit derivation from the parent theorem.

5.4 Proof Block

What it is: A formal argument establishing a theorem, lemma, or corollary.

Structure:

```
**Proof.** The argument establishing the claim. May span multiple
paragraphs and include display math, sub-lists, and case analysis.

The proof concludes with the end-of-proof marker on its final line. QED
```

The label `**Proof.**` begins the block. The end-of-proof marker `QED` MUST appear on the last line. The proof body MAY contain prose paragraphs, display math, and sub-lists. It MUST NOT contain headings, tables, or code blocks.

5.5 Scope Block

What it is: The document's declaration of what it covers.

Structure: Section 0 of the document body, with subsections for positive scope (what the document governs) and invariants (what the document commits to).

Every SPEC and FRAMEWORK document MUST include a Section 0 with scope and invariants declarations. Other document classes SHOULD include scope when the document's boundaries are not self-evident.

5.6 Negative Scope Declaration

What it is: An explicit statement of what the document does NOT address.

Structure: A prose paragraph or list within the Scope block, using the pattern:

```
This specification does not govern [topic]. That concern belongs
to [other document or future work].
```

Negative scope declarations are the sole permitted use of negative framing (see Section 2.1). They prevent scope creep by making boundaries explicit.

5.7 Invariants Block

What it is: A declaration of structural commitments the document makes.

Structure: A numbered list within Section 0, where each item states a property the document guarantees. Invariants MUST be verifiable claims – if an invariant cannot be checked, it is aspiration, not an invariant.

6. Composition and Nesting

This section declares which content types MAY appear inside which other content types, and the maximum nesting depth for each.

6.1 Containment Matrix

The following matrix declares valid containment relationships. Content types not listed as valid containers MUST NOT contain other content types.

Container	May contain
Prose paragraph	Inline math, inline code, bold, italic, bare Unicode symbols (isolated only)
List item (ordered or unordered)	Prose, inline math, inline code, bold, italic. Display math with caution (see 6.2). Code blocks with caution (see 6.2). No tables.
Display math	LaTeX math content only. No markdown formatting inside.
Code block	Literal text only. No formatting interpretation.
Table cell	Prose, inline code, inline math (short). No display math. No code blocks. No nested tables.
Definition block	Prose, inline math, display math.
Theorem/Lemma/Corollary block	Prose, inline math, display math.
Proof block	Prose, inline math, display math, sub-lists. No headings, tables, or code blocks.

6.2 Nesting Depth Limits

Maximum nesting depth for any container type: 3 levels.

Depth-specific rules for list items:

- Depth 1 (top-level list item): Display math and code blocks are permitted.
- Depth 2 (nested list item): Display math is permitted with caution – reduced width may cause overflow. Code blocks are permitted with caution.
- Depth 3 (double-nested list item): Display math MUST be extracted to a standalone block before or after the list. Code blocks MUST be extracted.

Extraction rule: If display math or a code block would appear inside a list item at nesting depth 3 or deeper, the author MUST extract it to a standalone block immediately before or after the enclosing list structure. The list item SHOULD reference the extracted content using prose.

6.3 Section Hierarchy

Headings MUST follow strict hierarchy: H2 inside H1, H3 inside H2, H4 inside H3. Heading levels MUST NOT be skipped (e.g., jumping from H2 directly to H4).

6.4 Block Adjacency

A blank line is REQUIRED above and below every block element. No exceptions. This includes:

- Before and after display math
- Before and after code blocks
- Before and after tables
- Before and after horizontal rules
- Between consecutive list blocks (a blank line between two separate lists)

Omitting blank lines between blocks causes Pandoc to merge or misinterpret adjacent elements.

7. Format Compatibility

This section declares the interaction status for every pairwise combination of format primitives.

7.1 Status Definitions

- SAFE. The combination works correctly in all contexts. No special handling is required.
- CAUTION. The combination works but has constraints. The specific constraint is noted. Authors SHOULD follow the noted guidance.
- PROHIBITED. The combination produces rendering failures. Authors MUST NOT use it. Preflight MUST flag it.

7.2 Compatibility Matrix

Inner element	In prose	In list item	In table cell	In bold/italic
Inline math `\(...\)`	SAFE	SAFE	SAFE (short)	SAFE
Inline code	SAFE	SAFE	CAUTION (width)	SAFE
Bold/italic	SAFE	SAFE	SAFE	N/A
Bare Unicode symbol (isolated)	SAFE	SAFE	SAFE	CAUTION
Bare Unicode expression (2+ ops)	PROHIBITED	PROHIBITED	PROHIBITED	PROHIBITED
Display math	SAFE	CAUTION (depth)	PROHIBITED	N/A
Code block	SAFE	CAUTION (depth)	PROHIBITED	N/A
Table	SAFE	PROHIBITED	PROHIBITED	N/A
Figure/image	SAFE	PROHIBITED	PROHIBITED	N/A
Long inline code (>60 chars)	PROMOTE	PROMOTE	PROHIBITED	PROHIBITED
Long URL (bare)	PROHIBITED	PROHIBITED	PROHIBITED	PROHIBITED

7.3 Rules Derived from the Matrix

The following rules are normative consequences of the compatibility matrix:

1) Math outside emphasis. Bold and italic SHOULD wrap prose words only. Math symbols and expressions inside emphasis markup MUST use inline math delimiters. Bare Unicode symbols inside bold or italic are CAUTION – they MAY interact unpredictably with emphasis parsing.
2) Display math depth limit. Display math inside list items is permitted at nesting depth 1. At depth 2, it is CAUTION (reduced line width). At depth 3 or deeper, it MUST be extracted to a standalone block (see Section 6.2).
3) Code block depth limit. Code blocks inside list items are permitted at nesting depth 1 and 2. At depth 3 or deeper, they MUST be extracted.
4) No tables in lists. Tables inside list items MUST NOT be used. The table MUST be extracted to a standalone block before or after the list.
5) No nested tables. Tables inside table cells MUST NOT be used.
6) Short content in table cells. Inline code and inline math in table cells are permitted but authors SHOULD keep content short. Effective cell width is limited. Inline code exceeding approximately 30 characters in a table cell is CAUTION.
7) URL formatting. All URLs MUST use markdown link syntax `[text](url)` or backtick inline code. Bare URLs in prose are PROHIBITED – they create unbreakable strings that overflow the page margin.
8) Inline code promotion. Inline code spans exceeding approximately 60 characters SHOULD be promoted to code blocks. This prevents overflow in narrow contexts (list items, table cells).
9) Bare expression prohibition. Two or more Unicode math symbols forming an expression MUST NOT appear as bare text. They MUST be wrapped in `\(...\)` inline math or `\[...\]` display math delimiters.

8. Content Requirements by Class

Each document class requires specific content elements. This section declares the minimum content surface for each class. Authors MUST include all required elements. Recommended elements SHOULD be included unless the document's scope makes them inapplicable.

8.1 SPEC

Required content:

- Scope declaration with invariants (Section 0)
- Definitions for all terms introduced
- Theorems with proofs for all formal claims
- SC-AS grounding traces for all definitions and constructs
- Cross-references resolving to existing sections or external documents

Recommended content:

- Applications section demonstrating the specification's use
- Appendices for reference tables, symbol registries, or extended examples

Pipeline stress: HIGH. Math-heavy content exercises the full compatibility surface.

8.2 FRAMEWORK

Required content:

- Scope declaration with invariants
- Derived constructs with definitions
- Convergence properties or structural guarantees
- SC-AS grounding traces
- Interface declarations (what the framework expects from and provides to other specifications)

Recommended content:

- Verification conditions or compliance tests
- Worked examples showing the framework in operation

Pipeline stress: HIGH. Math-heavy with dense format interactions.

8.3 PAPER

Required content:

- Abstract
- Introduction stating the problem and contribution
- Methodology or approach
- Results
- Discussion
- Conclusion
- References (bibliography)

Recommended content:

- Background or related work section
- Figures and illustrations
- Appendices for proofs, data, or extended analysis

Pipeline stress: HIGH. Academic formatting adds citations, figures, and bibliography processing.

8.4 MODULE

Required content:

- Scope declaration
- Module interface (what it exposes, what it depends on)
- Content appropriate to the module type (term ledger, verification ledger, etc.)

Recommended content:

- Generation method (how the module's content was produced)
- Resolution or maintenance policy

Pipeline stress: LOW to MEDIUM. Primarily tables and structured data.

8.5 RCC (Reflexive Closure Certificate)

Required content:

- Target identification (Document ID, version, SHA-256)
- Basis (specifications and versions the certificate covers)
- Verification conditions checked
- Result verdict (PASS, FAIL, or CONDITIONAL)

Recommended content:

- Findings index
- Assumptions
- Tooling notes

Pipeline stress: LOW. Fixed structure, minimal author variation.

8.6 LEDGER

Required content:

- Scope (what material the ledger governs)
- Ledger type declaration
- Entries in the declared format

Recommended content:

- Resolution classes
- Unresolved surface policy
- Generation method

Pipeline stress: LOW to MEDIUM. Primarily tables.

8.7 POLICY

Required content:

- Scope (what the policy applies to)
- Policy rules, stated as enforceable constraints
- Enforcement scope (who or what enforces)

Recommended content:

- Related artifacts
- Rationale for policy decisions

Pipeline stress: LOW. Prose-heavy.

8.8 NOTE

Required content:

- Topic identification
- Content appropriate to the note's purpose

Notes are non-normative by definition (`NOTE::NonNormative-Only: TRUE`). They carry no structural commitments and no formal requirements beyond coherent prose.

Pipeline stress: LOW.

9. Class Templates

Each document class has a recommended section ordering. These are skeletons, not rigid mandates – authors MAY adapt section ordering but MUST maintain logical dependency ordering (definitions before theorems, methodology before results).

9.1 SPEC Template

```
## 0. Scope and Invariants
## 1. Definitions
## 2–N. [Theorems and Results — one section per major result area]
## N+1. Applications (if applicable)
## Appendices
```

9.2 FRAMEWORK Template

```
## 0. Scope and Invariants
## 1. Foundations (derived constructs, grounding)
## 2–N. [Framework components — one section per major component]
## N+1. Convergence Properties / Structural Guarantees
## N+2. Interface Declarations
## Appendices
```

9.3 PAPER Template

```
Abstract
## 1. Introduction
## 2. Background / Related Work
## 3. Method
## 4. Results
## 5. Discussion
## 6. Conclusion
## References
## Appendices
```

9.4 MODULE Template

```
## 0. Scope
## 1. Interface
## 2–N. [Module content — varies by module type]
## Appendices
```


9.5 RCC Template

```
## Target and Scope
## Verification
## Result
```

9.6 GUIDE Template

```
## 0. Architecture
## 1–N. [Rules -- one section per rule category]
## N+1. Checklist
## Appendices
```

9.7 POLICY Template

```
## 0. Scope
## 1. Policy Rules
## 2. Enforcement
## Appendices
```

9.8 LEDGER Template

```
## 0. Scope
## 1. Ledger Entries
## Appendices
```

10. Content Ordering

This section declares ordering constraints that apply across all document classes. These are structural requirements – violation produces documents where the reader cannot follow the argument.

10.1 Definition Before Use

Every term that is formally defined MUST be defined before its first use in normative content. A forward reference to an undefined term is permitted only if:

- The term is introduced with italic emphasis at its first mention (e.g., *canonical conformance*), AND
- The definition appears within the same major section (H2 level)

Cross-document references to terms defined in dependency specifications (listed in `Canonical-Dependencies`) are permitted without local redefinition.

10.2 Scope Honoring

All content in a document MUST fall within the declared scope (Section 0). Out-of-scope claims are not permitted. If a document needs to make a claim outside its declared scope, the scope MUST be amended or the claim MUST be moved to an appropriate document.

10.3 Cross-Reference Integrity

All cross-references MUST resolve to existing sections within the document or to existing external documents identified by Document ID. Dangling references (pointing to sections or documents that do not exist) are conformance failures.

10.4 Progressive Disclosure

Content SHOULD be ordered from simple to complex:

- Base cases before recursive cases
- Definitions before theorems that use them
- Assumptions stated before relied upon
- Concrete examples before abstract generalizations

The reader SHOULD be able to follow the argument without forward references to concepts not yet introduced. Where this is not achievable (e.g., mutually recursive definitions), the author MUST note the forward dependency explicitly.

10.5 Dependency Ordering of Sections

Sections MUST be ordered so that each section depends only on sections that precede it. If Section 5 references a construct defined in Section 7, one of them MUST be reordered. Circular section dependencies indicate a structural problem in the document's organization.

11. Pipeline Interface

This section declares what authors need to know about the rendering pipeline. It is not a complete pipeline specification – that is the concern of SC-PIPE (future). This section establishes the author-facing contract: what the pipeline expects, what it produces, and where the boundaries are.

11.1 Rendering Chain

Source documents are rendered through the following chain:

```
Source markdown (.md)
  |  Pandoc parser
  v
Pandoc AST
  |  Lua filter (page breaks, horizontal rules, boxed math)
  v
LaTeX document
```

```
  |  xelatex engine
  v
PDF output
  |  Post-render validation (overflow check)
  v
Verified institutional artifact
```

Each stage has constraints that upstream content MUST satisfy. This specification (SC-DOC) constrains source content to what the pipeline can render. If SC-DOC permits a pattern that the pipeline cannot handle, that is a specification gap – not an author error.

11.2 What the Pipeline Expects

The pipeline expects source content conformant to this specification. Specifically:

- All math expressions properly delimited (Section 4.1)
- No prohibited format combinations (Section 7)
- Nesting within declared depth limits (Section 6)
- Blank lines between all block elements (Section 6.4)
- LF line endings throughout
- UTF-8 encoding
- Canonical header conformant to SC-HDR-000001

11.3 What the Pipeline Produces

Given conformant source, the pipeline produces:

- A PDF with consistent typography, margins, and page breaking
- Correct rendering of all declared Unicode symbols (Appendix A)
- Display math in boxed environments (via Lua filter)
- Automatic page breaks before H1 sections
- Horizontal rules converted to visual separators

11.4 Pipeline Limitations

The following limitations exist in the current pipeline. Authors MUST work within these constraints:

1) No footnotes. The pipeline does not support footnotes in SC-AS specifications. Parenthetical remarks or endnotes MUST be used instead.
2) No figure/image block. A formal figure block type with sizing is not yet defined. Images MAY be included using standard markdown image syntax (`![alt](path)`) but sizing and placement are not guaranteed.
3) Citation processing. Bibliography and citation processing (pandoc-citeproc) is not yet integrated. PAPER class documents requiring citations SHOULD coordinate with the pipeline maintainer.
4) Multi-format output. Only PDF output is currently supported. HTML and EPUB generation are future capabilities.

11.5 The L2-L3 Contract

The critical interface in the document pipeline is between this specification (L2, authoring grammar) and the rendering pipeline (L3, template and engine).

The contract: SC-DOC MUST NOT permit any source pattern that the pipeline cannot render cleanly. The pipeline declares its constraints. SC-DOC honors them.

If an author follows this specification and the output has rendering failures, the failure is in the pipeline – not the document. The correct response is a template issue against SC-PIPE, not a document correction.

12. Pipeline Compatibility Checklist

This checklist maps directly to the grammar. Each item corresponds to a normative rule in this specification. Preflight enforces all items marked with [P]. Items marked with [R] require human review.

- [P] Canonical header is the first content in the file (SC-HDR-000001)
- [P] H1 title appears after the extension block (3.2)
- [P] H1 has no trailing # (3.2)
- [P] No heading levels are skipped (6.3)
- [P] All `---` horizontal rules have blank lines above and below (3.8)
- [P] All list nesting uses 3-space indent, not 4-space (3.3, 3.4)
- [P] All display math uses `\[...\]` or `$...$` with blank lines above and below (3.5)
- [P] Display math content lines inside `\[...\]` use 3-space indent (3.5)
- [P] All code blocks have triple backtick fences and a language identifier (3.6)
- [P] All inline math uses `\(...\)` delimiters (4.1)
- [P] Mathematical expressions (2+ operators) use delimiters, not bare Unicode (4.1, 7.3.9)
- [P] Expressions with 3+ operators or >60 chars are promoted to display math (4.1)
- [P] No bare LaTeX commands in prose without `\(...\)` delimiters (4.1)
- [P] No double-backslash typos in LaTeX commands (4.1)
- [P] Blank lines between all block elements (6.4)
- [P] Nesting depth does not exceed 3 levels (6.2)
- [P] No display math at nesting depth 3+ (6.2)
- [P] No tables inside list items (7.3.4)
- [P] No bare URLs in prose (7.3.7)
- [P] Table column count does not exceed 4 for prose tables (3.7)
- [P] LF line endings throughout (SC-HDR-000001)
- [P] SHA-256 re-stamped after all changes (SC-HDR-000001)
- [R] Definition-before-use ordering (10.1)
- [R] Content stays within declared scope (10.2)
- [R] Cross-references resolve (10.3)
- [R] Progressive disclosure ordering (10.4)
- [R] Section dependency ordering (10.5)
- [R] Austerity: all constructs trace to SC-AS primitives (2.2)
- [R] Concept introduction follows PCF ordering (2.3)

Appendix A: Supported Unicode Math Symbols

These symbols MAY appear in prose without math delimiters when used as isolated symbols (not as part of expressions). The export template maps them to LaTeX equivalents via `\newunicodechar`.

Set theory: `\in` `\notin` `\emptyset` `\subseteq` `\supseteq` `\cup` `\cap`

Logic: `\forall` `\exists` `\land` `\lor` `\Rightarrow` `\Leftrightarrow` `\leftrightarrow` `\models` (and negation)

Ordering: `\neq` `\leq` `\geq` `\prec` `\preceq` `\precsim` `\sim`

Greek: `\varphi` `\varepsilon` `\delta` `\pi`

Brackets: `\langle` `\rangle`

Other: `\cdots` `\mathcal{L}` `\leadsto` `\triangleq`

If a document requires a Unicode math symbol not in this list, a `\newunicodechar` mapping MUST be added to `release/export/coherence.latex` before export. Unmapped Unicode symbols MUST NOT appear in source documents – they produce missing glyphs or compilation errors.

Appendix B: Failure Class Registry

The following table catalogs known failure modes, their location in the document surface, and their governance status under this specification.

ID	Description	Surface cell	Governed by	Preflight
FC-1	Bare math expressions (2+ operators without delimiters)	CF	4.1, 7.3.9	Yes
FC-2	Long bare URLs	CF	7.3.7	Yes
FC-3	Wide tables (5+ columns, long cells)	CF, PF	3.7, 7.3.6	Partial
FC-4	Display math in nested lists (depth 3+)	CS, CF	6.2, 7.3.2	Yes
FC-5	Long inline code spans (>60 chars)	CF	4.2, 7.3.8	Yes
FC-6	Wrong math delimiters (format string mismatch)	FF	4.1, 11.2	Partial
FC-7	Missing blank lines between blocks	CS	6.4	Yes
FC-8	Deep nesting (>3 levels)	CS	6.2	Yes
FC-9	Bold/italic containing math symbols	CF	4.3, 7.3.1	Partial
FC-10	Page break control (widows/orphans)	FS	11.3	No (L3)

ID	Description	Surface cell	Governed by	Preflight
FC-11	Figures/images (no formal block type)	PF	11.4	No (future)
FC-12	Inconsistent theorem/definition patterns	PS	5.1-5.4	Partial
FC-13	Footnotes	PF	11.4	Yes
FC-14	Special characters (bare ampersand, percent)	CF	(Pandoc handles)	No

Appendix C: PCF Structural Analysis

This specification was derived from a PCF analysis of the document as a system. The document surface maps to the PCF axes:

- Content = Presence. What semantic elements exist in the document.
- Structure = Connection. How those elements relate, compose, and nest.
- Format = Flow. How content and structure are expressed in source and transformed to output.

PCF applied to PCF yields the 9-cell document surface. Every cell that is ungoverned is a cell where failures emerge. This specification governs all 9 cells:

Cell	Concern	Sections
PC (Presence of Content)	What content elements each class requires	8
CC (Connection of Content)	How content elements reference each other	10
FC (Flow of Content)	How the argument builds coherently	9, 10.4
PS (Presence of Structure)	What structural containers exist	3, 4, 5
CS (Connection of Structure)	How containers compose and nest	6
FS (Flow of Structure)	How the document unfolds by class	9
PF (Presence of Format)	What formatting primitives exist and their expression	3, 4
CF (Connection of Format)	How formatting primitives interact	7
FF (Flow of Format)	How source transforms to output	11

The predecessor document (SC-AUTH-000001) governed cells PF and partial CF. This specification extends governance to the complete surface.

The design analysis that produced this specification is documented in CR-INSTITUTIONAL-PIPELINE-DESIGN (docs/internal/).

End of Document