Publishing Repos Operational Semantics Contract v0.7.1

1. Purpose

This contract defines the operational semantics for the Probabilistic Systems Engineering publishing repository.

Its purpose is to transform committed publishing inputs into a deployed archive site while preserving explicit authority over:

- publish-unit discovery
- HTML export normalization
- structural classification
- output topology
- homepage and document navigation behavior
- structured metadata emission
- published-state observability
- deployment and reconciliation behavior
- refusal conditions

This contract governs the publishing and deployment repository only.

Canonical prose-authoring authority remains outside this repository.

2. Scope

This contract applies to:

- papers

- contracts
- replication materials

This contract covers:

- committed incoming publish inputs
- grouped incoming folder traversal
- Google Docs HTML zip extraction and normalization
- rendered document generation
- homepage and per-document navigation generation
- per-document structured metadata emission
- site-level metadata index generation
- sitemap generation
- explicit reference discovery
- thresholded related-document recommendations
- deployment
- build manifest generation
- scheduled reconciliation and self-heal behavior

This contract does not yet cover:

- comments
- social features
- broad semantic search
- tags

- sidecar metadata files
- arbitrary author-curated taxonomies
- opaque embedding-based similarity
- Google Drive ingestion
- arbitrary multi-tab semantic navigation reconstruction beyond current normalization rules

3. Repository Roles

3.1 Canonical Authoring Source

Canonical document authority remains outside this repository.

The repository SHALL NOT be treated as the prose authoring source of truth.

3.2 Repository Role

The repository is the publishing and deployment system.

It stores:

- committed publish inputs
- build scripts
- templates
- workflow definitions
- minimal site scaffolding
- optional legacy PDF-only contract artifacts where explicitly supported by this contract

3.3 Generated Output

Generated site output is derived state only.

`dist/` SHALL be treated as build output.

Generated HTML, extracted assets, metadata artifacts, and `dist/` contents SHALL NOT be committed back into git as part of normal publishing behavior.

4. Source Artifact Model

4.1 Incoming Roots

Committed publish inputs SHALL exist under:

- `incoming/papers/`
- `incoming/contracts/`
- `incoming/replication/`

4.2 Grouping Folders

Grouping folders MAY exist at arbitrary depth under an incoming root.

Grouping folders are organizational only.

A grouping folder SHALL NOT itself be treated as a publishable unit unless it independently satisfies §4.3.

4.3 Publish Unit

A renderable publish unit is a directory under an incoming root that contains:

- exactly one `*.pdf`
- exactly one `*.zip`

A directory that does not satisfy both requirements SHALL NOT be published.

4.4 Publish Unit Identity

The publish-unit identity SHALL be its relative path from the incoming type root.

Examples:

- `incoming/papers/foo/` → slug `foo`
- `incoming/papers/program-a/paper-1/` → slug `program-a/paper-1`
- `incoming/replication/context-injection/glossary-v0.7/` → slug `context-injection/glossary-v0.7`

The full relative slug path is authoritative for output routing and grouping.

4.5 Display Label

Default display label SHALL be the PDF stem.

No sidecar metadata file is required in v0.7.1.

5. Discovery Semantics

5.1 Traversal

Discovery SHALL traverse each incoming type root recursively.

Traversal order SHALL NOT affect output semantics.

5.2 Candidate Directory Rule

If a directory contains any governed publish artifacts (`*.pdf` or `*.zip`), that directory MUST be treated as a candidate publish unit.

5.3 Qualification and Refusal

For a candidate publish-unit directory:

- if it contains exactly one PDF and exactly one ZIP, it SHALL qualify as a publish unit
- otherwise build SHALL fail for that directory
- the system SHALL NOT guess intended grouping or descend further in search of a fix

5.4 Traversal Termination

If a directory qualifies as a publish unit, traversal below that directory SHALL terminate.

Child directories beneath a qualified publish unit SHALL NOT be independently discovered or published.

6. HTML Zip Handling

6.1 Accepted Export Shape

The system SHALL accept Google Docs HTML exports packaged as zip files.

6.2 Extraction Rule

For each publish unit, the zip SHALL be extracted into a temporary working directory.

6.3 Main HTML Detection

After extraction:

- if exactly one `.html` file exists anywhere in the extracted tree, it SHALL be used as the main document
- if zero `.html` files exist, build SHALL fail
- if more than one `.html` files exist, build SHALL fail

The system SHALL NOT guess a primary HTML file.

6.4 Asset Preservation

Assets required for correct rendered output SHALL be preserved in published output.

7. Normalization Semantics

7.1 Head and Metadata Normalization

Rendered pages SHALL normalize or inject:

- page title
- page description
- author metadata

- canonical URL
- per-page structured metadata

7.2 Exported Style Preservation

Exported Google Docs `<style>` blocks MAY be preserved where required for text fidelity.

7.3 Empty Paragraph Cleanup

Paragraphs containing no meaningful text and no structural content SHALL be removed.

Paragraphs containing structural content such as images, tables, SVG, or rules SHALL NOT be treated as empty.

7.4 Generic Paragraph Class Cleanup

Generic Google Docs structural classes of the form `cN` on normal `<p>` elements SHALL be stripped during normalization.

Pipeline-specific semantic classes MAY be preserved.

7.5 Generic List-Item Class Cleanup

Generic Google Docs structural classes of the form `cN` on `<li>` elements SHALL be stripped during normalization.

List-marker support classes required for correct marker rendering SHALL be preserved.

7.6 Title Canonicalization

If multiple title-class paragraphs exist in exported content, normalization SHALL resolve them to exactly one canonical rendered title block.

The canonical title SHALL be selected by this algorithm:

1. collect all non-empty title-class paragraph texts in document order
2. normalize text for comparison by trimming whitespace and decoding entities
3. count occurrences of each normalized title text
4. select the title with the greatest frequency

5. if multiple titles tie on frequency, select the tied title with the greatest normalized text length
6. if multiple titles still tie after length comparison, select the lexicographically smallest normalized text
7. preserve only the first occurrence of the winning title in document order
8. remove all other title-class paragraphs

The system SHALL NOT preserve multiple competing top-level title paragraphs.

7.7 Classification Semantics

The build MAY classify normalized paragraphs into pipeline-specific display classes, including:

- `pse-callout`
- `pse-lead-in`
- `pse-compact`

Classification is heuristic, build-time, and output-facing.

7.8 Boundary-Scoped Classification

Classification inheritance SHALL NOT cross structural boundaries.

The following SHALL reset callout inheritance:

- `hr`
- `h1`
- `h2`
- `h3`
- `h4`

- `h5`
- `h6`
- `ul`
- `ol`

7.9 Lead-In Classification

A paragraph SHALL be classified as `pse-lead-in` if all of the following are true:

- normalized text ends with `:`
- normalized text length is less than or equal to 140 characters
- the paragraph is not title content
- the paragraph is not subtitle content
- the paragraph is not list content
- the paragraph is not already classified as `pse-callout`

7.10 Callout Classification

A paragraph SHALL be classified as `pse-callout` only if all of the following are true:

- normalized text length is greater than or equal to 110 characters
- normalized text length is less than or equal to 420 characters
- the paragraph is not title content
- the paragraph is not subtitle content
- the paragraph is not list content
- the previous visible paragraph exists

- the previous visible paragraph's normalized text ends with `:`
- no structural boundary from §7.8 occurs between the previous visible paragraph and the candidate paragraph

A paragraph SHALL NOT become `pse-callout` solely because an earlier paragraph elsewhere in the document ended with `:`.

7.11 Compact Classification

A paragraph SHALL be classified as `pse-compact` only if all of the following are true:

- normalized text length is less than or equal to 95 characters
- the paragraph is not title content
- the paragraph is not subtitle content
- the paragraph is not list content
- the paragraph is not `pse-lead-in`
- the paragraph is not `pse-callout`

8. Output Topology

8.1 Deploy Artifact Root

The build SHALL generate deployable output under `dist/`.

8.2 Document Output Paths

For each valid publish unit, the system SHALL generate:

- `dist/<type>/<relative-slug>/index.html`
- `dist/<type>/<relative-slug>/<pdf-filename>`
- extracted assets under `dist/<type>/<relative-slug>/...`

Where:

- `<type>` is `papers`, `contracts`, or `replication`
- `<relative-slug>` preserves the full relative incoming path

8.3 HTML Entry Name

The rendered HTML entry page for every rendered document SHALL be named `index.html`.

8.4 Sitemap Artifact

The build SHALL generate `dist/sitemap.xml`.

It SHALL include:

- the homepage URL
- rendered HTML document URLs

PDF URLs are not required in sitemap scope for v0.7.1.

8.5 Site Metadata Index Artifact

The build SHALL generate `dist/metadata/documents.json`.

It SHALL contain:

- `site`
- `site_url`
- `generated_at_utc`
- `documents`

There SHALL be one metadata entry per rendered document.

9. Legacy PDF-Only Contract Semantics

9.1 Supported Legacy Mode

Legacy PDF-only contract entries stored under `contracts/` SHALL be surfaced on the homepage only when no rendered incoming contract with the same slug exists.

9.2 PDF-Only Entry Behavior

A PDF-only legacy entry:

- SHALL link directly to the PDF as its primary title target
- SHALL expose PDF action semantics only
- SHALL NOT imply the existence of a rendered HTML page

9.3 Precedence

If both a rendered incoming contract and a legacy PDF-only contract exist for the same slug, the rendered incoming contract SHALL take precedence and the PDF-only legacy entry SHALL NOT be separately surfaced.

10. Homepage and Index Semantics

10.1 Homepage Purpose

The homepage is an archive landing page.

10.2 Homepage Contents

`dist/index.html` SHALL contain:

- archive/site title
- author attribution
- archive framing text
- Start Here guidance
- a Papers section
- a Contracts section

- a Replication Materials section

10.3 Homepage Layout Semantics

Homepage section layout SHALL preserve Papers, Contracts, and Replication Materials as distinct top-level sections.

10.4 Section Ordering

Top-level homepage sections SHALL be ordered:

1. Papers
2. Contracts
3. Replication Materials

10.5 Grouping Semantics

Entries SHALL be grouped by the first slug segment when the slug contains more than one segment.

Entries whose slug contains only one segment SHALL render directly in their parent content section without a synthetic subgroup heading.

10.6 Group Presentation

Group headings SHALL be derived from slug structure.

Synthetic headings such as “Independent Documents” SHALL NOT be introduced for flat items.

10.7 Group Ordering

Within each top-level content section:

- flat items SHALL render before grouped blocks
- grouped-block ordering SHALL be deterministic

10.8 Item Ordering

Within flat items and within each grouped block, item ordering SHALL be deterministic and SHALL remain subject to version-family collapse rules in §13.

10.9 Entry Semantics

Each entry SHALL expose a valid primary click target.

For rendered entries:

- the main title SHALL link to the rendered document page
- explicit actions SHALL include Read and PDF

For PDF-only entries:

- the main title SHALL link to the PDF
- explicit actions SHALL reflect PDF-only status
- no dead HTML target SHALL be implied

10.10 Reachability Invariant

Every homepage entry SHALL expose at least one valid navigable artifact.

11. Structured Metadata Semantics

11.1 Metadata Derivation Source

Per-document structured metadata SHALL be derived from existing HTML/PDF-derived document context.

Sidecar metadata files are out of scope in v0.7.1.

11.2 Required Per-Document Metadata Fields

Each rendered document SHALL have derived metadata including at least:

- `kind`
- `schema_type`

- `content_type`
- `slug`
- `title`
- `author`
- `html_path`
- `html_url`
- `pdf_path`
- `pdf_url`
- `group_key`
- `family_key` when applicable
- `version`
- `version_tuple` when applicable
- `description`
- `word_count`
- `reading_time_minutes`

11.2A Derived Field Rules

Derived metadata fields SHALL be produced from existing HTML/PDF-derived document context.

If a field cannot be derived safely, omission or empty value is preferred over speculative fabrication.

11.3 Page-Level Metadata Emission

Each rendered document page SHALL emit:

- canonical URL
- JSON-LD structured metadata

12. Discovery Semantics

12.1 Discovery Surfaces

The system MAY emit two distinct document-page discovery surfaces:

- Referenced artifacts
- Read next

These surfaces are distinct and SHALL NOT be merged into one unlabeled section.

12.2 Referenced Artifacts

Referenced artifacts are grounded links derived from explicit document-text evidence.

12.3 Explicit Reference Signals

Explicit reference detection MAY use:

- normalized title phrase match
- slug-derived phrase match
- version + strong token support

12.4 Explicit Reference Conservatism

Referenced-artifact detection SHALL prefer omission over weak or ambiguous matches.

False positives are worse than empty output.

12.5 Read Next

Read next is a heuristic recommendation surface derived from deterministic scoring.

12.6 Related-Docs Scoring Inputs

Related-document scoring MAY use signals such as:

- explicit reference relationship
- group relationship
- content-type relationship
- title similarity
- description similarity
- limited cross-kind affinity, at minimum for:
    - paper ↔ contract
    - paper ↔ replication

12.7 Recommendation Limits

Read-next recommendations SHALL:

- be capped at a maximum of 2 entries
- be suppressed entirely when confidence is weak
- exclude the current document
- remain distinct from Referenced artifacts

13. Version and Lineage Semantics

13.1 Versioned Slug Identity

A versioned slug is the immutable published artifact identity.

Published versioned URLs SHALL remain stable.

13.2 Slug-Family Derivation

Version-family derivation SHALL use only the final slug segment.

13.3 Version Suffix Pattern

A slug segment SHALL be treated as versioned only if it matches a terminal version suffix pattern of the form:

- `-v1`
- `-v1.2`
- `-v1.2.3`

Equivalent regex shape:

- `^(?P<family>.+)-v\d+(?:\.\d+)*$`

13.4 No Broader Family Inference

The system SHALL NOT infer version families from titles, descriptions, or broader semantic similarity in v0.7.1.

13.5 Latest-Only Index Visibility

When multiple versions exist in the same slug family:

- only the latest version SHALL appear in the main homepage index
- older versions SHALL remain reachable by direct URL

13.6 Latest Determination

Latest version SHALL be determined by parsed version value from the final slug segment, not by lexical string ordering alone.

13.7 Page-Level Version Navigation

Rendered document pages MAY surface version relationship guidance derived from slug-family relationships only.

13.8 No Historical Relocation Requirement

Older versions SHALL NOT be relocated into separate archival subtrees as part of normal version handling.

14. Document Navigation Semantics

14.1 Home Navigation

Per-document home navigation SHALL resolve correctly for the document's relative slug depth.

The system SHALL derive home-link depth from slug structure rather than using a hardcoded constant.

14.2 PDF Navigation

Per-document PDF navigation SHALL resolve to the PDF artifact in the same published output directory.

15. Build Manifest and Published State

15.1 Manifest Emission

A successful build SHALL emit `dist/build.json`.

15.2 Required Manifest Fields

`dist/build.json` SHALL contain:

- `source_sha`
- `source_ref`
- `built_at_utc`
- `dist_hash`

15.2B dist_hash Role

`dist_hash` is emitted for observability of generated published state and is not required as a scheduled reconciliation gate.

15.3 Observability Invariant

Published state SHALL be externally observable without requiring inference from Git history alone.

16. Deployment and Reconciliation Semantics

16.1 Deployment Mechanism

Deployment SHALL use the repository's configured publish and hosting mechanism.

16.2 Deploy Source

The deploy source SHALL be the generated `dist/` artifact.

16.3 Normal Publish Paths

The publish workflow MAY run on normal source-change and manual invocation paths.

16.4 Scheduled Reconciliation

Scheduled reconciliation SHALL compare live published state to current intended source state.

16.5 Drift Detection

Drift SHALL include:

- missing or unreadable live `build.json`
- live `build.json.source_sha` missing or empty
- live `build.json.source_sha` differing from the current source state for the run

16.6 Self-Heal Policy

Scheduled reconciliation SHALL republish on detected drift.

16.7 Missing Live Manifest

If live `build.json` is missing or unreadable during scheduled reconciliation, that condition SHALL be treated as drift and self-heal republish is permitted.

16.8 Non-Mutation Invariant

Reconciliation and self-heal behavior SHALL NOT mutate repository source artifacts or commit generated output back into git.

17. Failure Model

17.1 Publish Unit Qualification Failures

Build MUST fail if any candidate publish-unit directory violates artifact cardinality:

- no PDF found
- more than one PDF found
- no ZIP found
- more than one ZIP found

17.2 Extraction Failures

Build MUST fail if:

- zip extraction fails
- no HTML file exists after extraction
- more than one HTML file exists after extraction

17.3 Output Construction Failures

Build MUST fail if:

- output directory cannot be constructed
- PDF copy into `dist/` fails
- homepage generation fails
- document page generation fails

- metadata index generation fails
- sitemap generation fails
- build manifest generation fails

17.4 Refusal Rule

When input shape is ambiguous, the system SHALL fail rather than guess.

17.5 Discovery Omission Rule

When discovery or recommendation confidence is insufficient, omission is valid and preferred to low-confidence output.

18. Invariants

INV-001 Publish Unit Qualification
A renderable publish unit MUST resolve to exactly one PDF and exactly one ZIP.

INV-002 Leaf Authority
Only qualified publish-unit directories are publishable units.

INV-003 Traversal Termination
Traversal MUST terminate below a qualified publish unit.

INV-004 Relative Slug Preservation
Output routing MUST preserve the full relative incoming slug path.

INV-005 Single HTML Entry
Each rendered publish unit MUST emit exactly one HTML entry page at `index.html`.

INV-006 Stable PDF Reachability
Each rendered publish unit MUST emit exactly one linked PDF artifact in its output directory.

INV-007 No Generic Docs Spacing Leakage
Generic Google Docs `cN` structural classes MUST NOT remain authoritative on normalized paragraphs or list items.

INV-008 Single Marker System
Normalized list rendering MUST NOT produce duplicate marker systems.

INV-009 Single Canonical Title
At most one canonical top-level title block MAY survive in rendered body content.

INV-010 Boundary-Scoped Classification
Callout inheritance MUST NOT cross structural boundaries.

INV-011 Fixed Classification Thresholds
Lead-in, callout, and compact paragraph eligibility MUST follow the exact thresholds defined in §7.9–§7.11.

INV-012 Index Reachability
Every homepage entry MUST expose a valid primary navigation target.

INV-013 Deterministic Homepage Ordering
Homepage section, group, and item ordering MUST be deterministic and MUST respect applicable version-family rules.

INV-014 Published State Observability
Successful builds MUST emit a manifest sufficient for live published-state observability.

INV-015 Self-Heal Without Source Mutation
Scheduled reconciliation MAY republish generated site output but MUST NOT mutate repository source artifacts.

INV-016 Legacy Contract Precedence
Rendered incoming contracts take precedence over PDF-only legacy entries with the same slug.

INV-017 Supported Content Types
Supported content types for this contract are papers, contracts, and replication.

INV-018 Sitemap Reachability
Sitemap entries MUST resolve to valid published HTML pages.

INV-019 Per-Document Metadata Emission
Every rendered document MUST emit the required derived metadata and page-level structured metadata.

INV-020 Site Metadata Index Completeness
`dist/metadata/documents.json` MUST contain one metadata entry per rendered document.

INV-021 Explicit Reference Conservatism
Referenced-artifact detection MUST prefer omission over weak or ambiguous matches.

INV-022 Related-Docs Conservatism
Related-doc recommendations MUST be thresholded, capped, and suppressible; false positives are worse than no recommendation.

INV-023 Slug-Family Authority
Version-family derivation MAY only use the final slug segment and terminal version suffix pattern.

INV-024 Stable Historical Reachability
Versioned artifact URLs remain stable even when newer versions exist.

INV-025 Latest-Only Index Visibility
When multiple versions exist in the same slug family, only the latest version appears in the main index.

INV-026 Depth-Correct Navigation
Per-document home navigation MUST resolve correctly for the document's relative slug depth.

INV-027 Reference and Recommendation Separation
Referenced artifacts and heuristic related-doc recommendations MUST remain distinct output surfaces.

INV-032 Scheduled Reconcile Commit Identity
Scheduled reconciliation MUST determine drift from live manifest readability and `source_sha` equality against the current source state for the run.

19. Non-Goals

This contract does not yet require:

- comments or social features
- sidecar metadata files or arbitrary taxonomy systems
- full-text or embedding-based search
- pixel-perfect fidelity to Google Docs layout
- repo-stored generated HTML trees
- historical relocation of older versions

20. Acceptance Criteria

This contract is satisfied when the system can:

1. consume committed grouped or flat incoming inputs under `incoming/papers/`, `incoming/contracts/`, and `incoming/replication/`
2. qualify only candidate directories containing exactly one PDF and one ZIP
3. terminate traversal below qualified publish units
4. extract and normalize Google Docs HTML export content
5. remove generic Docs structural spacing leakage from normalized paragraph/list layout
6. canonicalize competing title-class paragraphs down to one rendered title using §7.6
7. classify lead-in, callout, and compact paragraphs using §7.9–§7.11
8. generate rendered pages under `dist/papers/...`, `dist/contracts/...`, and `dist/replication/...`
9. generate a homepage with Papers, Contracts, and Replication Materials and deterministic ordering
10. emit per-document structured metadata and site-level metadata index artifacts
11. emit `sitemap.xml`
12. emit `build.json` with the required fields
13. detect explicit references conservatively and emit them when confidence is sufficient
14. emit at most two related-doc recommendations and suppress them when confidence is weak
15. preserve old versioned URLs while showing only latest versions in the main index
16. derive version families only from the final slug segment
17. resolve per-document home navigation correctly for document depth
18. deploy from `dist/`
19. detect drift via scheduled reconciliation and republish on drift
20. fail explicitly rather than guess when input shape or discovery confidence is ambiguous