Artifact Projection — Convergence Contract v2.4.1

Date: March 1, 2026
Status: DRAFT (DS Stabilization: engine selection; engine preflight; extracted-text canonicalization; schema alignment)

1. Purpose

This contract defines convergence semantics for a repository-resident projection system that derives Markdown artifacts from canonical PDF sources.

The system MUST:

1. Derive all decisions exclusively from repository state at the triggering commit (“C_after”).
2. Compute convergence against a logically defined final topology.
3. Stage all mutations in an isolated workspace prior to repository-visible changes.
4. Guarantee atomic repository-visible updates.
5. Prevent mutation of unmanaged files.
6. Enforce deterministic mapping from source artifacts to derived units.

This contract defines decision surfaces and authority boundaries. It does not prescribe implementation strategy.

2. Conceptual Model (Normative Vocabulary)

2.1 Authority Domain

The Authority Domain is the set of inputs that determine desired state. In this contract: eligible PDF source artifacts at C_after.

2.2 Deterministic Projection

A pure function mapping each authoritative input to one or more deterministic output paths.

2.3 Deterministic Target Set (DTS)

The complete set of all deterministic output paths computed from the Authority Domain.

2.4 Managed Residue

Paths under projection control that are not part of the valid final deterministic state but are subject to projection deletion authority.

2.5 Logical Final Managed State (LFMS)

The repository state in which:

1. Every DTS unit exists and is valid.
2. No Managed Residue exists.
3. Unmanaged files remain untouched.

2.6 Decision Surface (DS)

Any contract rule whose interpretation can change externally observable outcomes, including:

1. Create vs not create
2. Delete vs retain
3. Halt vs proceed
4. Success vs failure classification

Stability requires DS determinism.

3. Inputs

3.1 Repository State

The sole authoritative input is the repository tree at commit C_after.

3.2 Configuration

Each Source Root defines:

1. `root_id` (string, unique)
2. `root_path` (repository-relative path)
3. `recursive` (boolean)

Required run-level keys:

1. `engine_kind` (string) — normative selector for conversion behavior (see §3.3)
2. `engine_id` (string) — stable identifier for the projection engine configuration (see §8.3 and §13)

3.3 Engine Selection (DS Stabilized)

1. `engine_kind` is the sole selector of which conversion path is executed.
2. `engine_id` MUST NOT be parsed to choose behavior.
3. `engine_kind` comparisons are byte-for-byte string equality. No normalization is implied.
4. `engine_kind` MUST be one of:
    1. `pymupdf_text`
    2. `pdf2htmlex_layout_html`
5. If `engine_kind` is missing or not one of the allowed values, the outcome MUST be `FAILED_POLICY`.

3.4 Engine Preconditions (DS Stabilized)

Before staging any repository-visible mutation, the system MUST perform a preflight check for the selected `engine_kind`.

Preflight MUST verify required dependencies are available:

1. For `engine_kind=pymupdf_text`: importing module `fitz` (PyMuPDF) succeeds.
2. For `engine_kind=pdf2htmlex_layout_html`: executables `pdf2htmlEX` and `pandoc` are present on PATH.

If any required dependency is missing or cannot be invoked/imported, the outcome MUST be `FAILED_OPERATIONAL`. Preflight failure MUST guarantee no repository-visible change.

4. Path Semantics and Root Constraints

4.1 Path Equality

All path comparisons are byte-for-byte, case-sensitive, repository-relative. No OS normalization is implied.

4.2 Segment-Prefix (Normative Definition)

For repository-relative paths A and B, A is a segment-prefix of B iff:

1. B == A, or
2. B starts with the bytes A + “/”.

Used for both root non-overlap and eligibility membership.

4.3 Root Path Constraints

Each `root_path` MUST:

1. Be non-empty.
2. Not start or end with “/”.
3. Not contain “//”.
4. Not contain “.” or “..” path segments.
5. Not equal “markdown”.
6. Not be a segment-prefix of `markdown` and not have `markdown` as a segment-prefix.
7. Not overlap any other `root_path` by segment-prefix.
8. Have unique `root_id` and unique `root_path`.

Violation is `FAILED_POLICY`.

5. Eligibility

A file is eligible iff:

1. It resides under exactly one configured Source Root.
2. Its filename ends with `.pdf` (case-sensitive).
3. Recursive rules apply:

    1. `recursive=true` → any depth under `root_path`.
    2. `recursive=false` → immediate parent directory equals `root_path`.

5.1 Resides Under Root

A file at `source_path` resides under a root with `root_path` iff:

1. `root_path` is a segment-prefix of `source_path`, and
2. `source_path != root_path`.

Matching zero roots → not eligible. Matching multiple roots → `FAILED_POLICY`.

5.2 Equality Semantics (Non-Recursive)

Eligibility for `recursive=false` holds iff:

1. `parent_dir(source_path)` is byte-for-byte identical to `root_path`.
2. Case-sensitive comparison.
3. No realpath normalization is implied.

6. Canonical Identity

Each eligible artifact is uniquely identified by (`source_path`, `root_id`). Changing `root_id` is an identity change.

7. Deterministic Mapping

For each eligible artifact:

1. `output_rel_path` is the relative path beneath the Source Root with the final `.pdf` suffix removed.
2. Projection outputs are:

    1. `markdown/<output_rel_path>.md`
    2. `markdown/<output_rel_path>.meta.json`

Mapping is recomputed every run.

8. Metadata Schema v2.4.0

Each `.meta.json` MUST contain exactly:

1. `schema_version` = `"2.4.0"`
2. `source_path`
3. `root_id`
4. `pdf_sha256` (lowercase hex SHA-256)
5. `engine_kind`
6. `engine_id`

No additional keys allowed.

8.1 Invalid Meta Is Fatal

If any `.meta.json` violates schema:

1. Execution MUST halt with `FAILED_POLICY`.
2. No convergence attempt occurs.
3. Invalid meta is NOT residue.

8.2 engine_id Semantics (DS Stabilized)

1. `engine_id` is REQUIRED in configuration and REQUIRED in every `.meta.json`.
2. `engine_id` comparisons are byte-for-byte string equality. No normalization is implied.

3. `engine_id` MUST change when the operator deliberately changes any behavior that could alter deterministic Markdown output, including:

    1. conversion engine configuration
    2. text extraction engine
    3. rendering engine

8.3 engine_kind Semantics (DS Stabilized)

1. `engine_kind` is REQUIRED in configuration and REQUIRED in every `.meta.json`.
2. `engine_kind` comparisons are byte-for-byte string equality. No normalization is implied.
3. `engine_kind` reflects the normative behavior family (selection surface), while `engine_id` captures specific configuration/version.

9. Managed Domain Model

All paths under `markdown/` fall into exactly one class.

9.1 Managed Final Unit (MFU)

A unit is MFU iff:

1. Both deterministic `.md` and `.meta.json` exist.
2. Metadata is schema-valid.
3. Metadata corresponds to an eligible artifact.
4. Mapping matches deterministic computation.

9.2 Managed Residue (MR)

MR includes:

1. MR-1 Partial pair at DTS path
2. MR-2 Mis-mapped valid meta
3. MR-3 Non-eligible or orphan meta case

9.3 Unmanaged

Any path under `markdown/` not classified as MFU or MR is Unmanaged. Unmanaged files MUST NOT be modified or deleted.

10. Convergence Sets

1. DesiredSet: all eligible artifacts at C_after.
2. DTS: deterministic output paths for DesiredSet.
3. LRS: Managed Residue paths.
4. LFMS: DTS exists, LRS empty, Unmanaged untouched.

11. Collision Evaluation

Detect:

1. Exact path equality collisions.
2. ASCII case-fold collisions (A–Z only).

Collision → `FAILED_POLICY`.

12. Strict Deletion Model

1. If `ARTIFACT_PROJECTION_ALLOW_DELETIONS != "true"` and LRS is non-empty → `FAILED_POLICY`.
2. If deletions are enabled → all LRS MUST be removed.

13. Regeneration Rules

Regenerate a DTS unit if any of the following differ from its current meta:

1. `schema_version`
2. `pdf_sha256`
3. `root_id`
4. `source_path`
5. `engine_kind`
6. `engine_id`
7. deterministic mapping path

No other triggers.

14. Representation Invariants

Markdown MUST:

1. Preserve heading order.
2. Preserve non-sequential heading levels.
3. Preserve list semantics.

14.1 Canonical Markdown Output (DS Stabilized)

1. Output MUST be GitHub-Flavored Markdown (GFM).
2. Structural/layout HTML is prohibited as canonical output.
3. If conversion output contains structural/layout HTML blocks intended to represent document structure (e.g., layout `div/span/img` used to represent the page), the outcome MUST be `FAILED_REPRESENTATION`.

14.2 Extracted Text Canonicalization (engine_kind=pymupdf_text) (DS Stabilized)

For `engine_kind=pymupdf_text`, extracted text MUST be canonicalized prior to final markdown emission such that list and heading semantics are preserved under GFM parsing. Minimum required canonicalization:

1. Replace CRLF and CR with LF.
2. Remove U+00AD (soft hyphen).
3. Remove zero-width characters, at minimum: U+200B, U+200C, U+200D, U+FEFF, U+2060.
4. Any line beginning with common bullet glyphs, including "●" and "•", MUST be rendered as a Markdown list item beginning with `-` .
5. If a line ends with ":" and the next non-empty line begins a list item, the output MUST include a blank line between the ":" line and the list block.
6. A line matching the pattern `^\d+(\.\d+)*\.\s+<title>` MUST be rendered as an ATX heading line in Markdown.
7. Headings MUST be separated from adjacent paragraphs by blank lines such that headings and lists do not become inline text.
8. Failure to satisfy §14.2 when the source content contains the triggering patterns is `FAILED_REPRESENTATION`.

14.3 Table Handling

If structured table extraction fails OR the extraction engine does not support structured table detection:

1. Insert `<!-- ARTIFACT_PROJECTION_DEGRADED -->`
2. Follow with a fenced code block containing raw table text.

14.4 Instability Scope (DS Clarified)

Representation variance that does not alter Decision Surface SHALL NOT constitute instability.

15. Staging and Atomicity

1. All mutations occur in isolated staging workspace.
2. Repository-visible state changes only after validation.
3. Commit is atomic.
4. Failure leaves no visible change.

16. Pre-Commit Validation

Before commit:

1. Engine preflight passed (§3.4).
2. No invalid meta exists.
3. LRS empty.
4. All DTS units exist as MFUs.
5. No Unmanaged modified.
6. Deterministic mapping verified.

Violation → `FAILED_POLICY` or `FAILED_OPERATIONAL`.

17. Self-Trigger Guard

1. If all ChangedPaths are under `markdown/` → exit without convergence.
2. If any non-markdown path changed → proceed.
3. Mixed commits proceed.

18. Outcomes

Exactly one:

1. `EXPORTED_CLEAN`
2. `FAILED_OPERATIONAL`
3. `FAILED_REPRESENTATION`
4. `FAILED_POLICY`

18.1 Outcome Classification Clarification (DS Stabilized)

1. Missing dependency / failed preflight (§3.4) → `FAILED_OPERATIONAL`.
2. Conversion completes but violates §14 representation invariants → `FAILED_REPRESENTATION`.
3. Configuration invalid per §3–§5, schema invalid per §8, or other policy violations → `FAILED_POLICY`.

19. Non-Goals

This contract does not guarantee:

1. OCR fidelity
2. pixel-perfect layout
3. exact PDF rendering equivalence
4. non-PDF source support

20. Fixture Suite

Fixtures MUST exist under `tools/artifact-projection/fixtures/`.

Each fixture MUST include:

1. `repo/`
2. `config.json`
3. `expect.json`
4. `expected_repo/`

`expect.json` MUST contain:

1. `expected_outcome` ∈ {`EXPORTED_CLEAN`, `FAILED_OPERATIONAL`, `FAILED_REPRESENTATION`, `FAILED_POLICY`}
2. `expect_changed_paths`
3. `expect_unchanged_paths`

Failure to match `expected_repo/` → `FAILED_REPRESENTATION`.

21. Required Fixtures

Must include:

1. Minimal Success Fixture
2. Non-Recursive Eligibility Fixture
3. Multi-Root Ambiguity Fixture
4. DTS Collision Fixture
5. DTS vs Unmanaged Fixture
6. Managed Residue Deletion Gate Fixture
7. Managed Residue Deletion Fixture
8. Regeneration Rule Fixture
9. Table Degradation Marker Fixture (engine_id pinned)

New (Engine DS Stabilization Fixtures):
10. Engine Selection Fixture (engine_kind pinned)

1. For `engine_kind=pymupdf_text`, expected output MUST NOT contain structural/layout HTML blocks.
2. Engine Preconditions Fixture
3. When the selected engine_kind dependency is unavailable at runtime, expected outcome MUST be `FAILED_OPERATIONAL`.
4. List Boundary Fixture
5. Source contains “MUST:” followed by bullet glyph list; expected output MUST be a real Markdown list block.
6. Heading Boundary Fixture
7. Source contains “2.1. Title …” pattern; expected output MUST render the heading as a heading, not inline paragraph text.

22. CI Deletion Safety

1. CI MUST run fixtures with deletions disabled.

2. CI MUST enable deletions only in an explicit deletion validation job.

23. Fixture Harness Requirements

Harness MUST:

1. Run against `repo/` as an isolated working tree.
2. Apply `config.json`.
3. Assert outcome equals `expect.json`.
4. Byte-for-byte compare `expected_repo/` when success.
5. Assert no repository-visible change on failure.
6. Classify any mismatch as `FAILED_REPRESENTATION`.

23.1 Operational Fixture Capability (Normative)

Harness MUST be capable of executing the Engine Preconditions Fixture such that dependency unavailability is real (not simulated by string-matching) and the observed outcome is `FAILED_OPERATIONAL`.