DOCUMENTATION PAGE

Page Registry Synchronization

Page Registry Synchronization

Use this procedure when adding one semantic Markdown page. The page file, its metadata header, exactly one object in `registries/pages.json`, the registered public-safe sources, and the generated projection form one synchronization unit. A readable page alone is not a complete product change: identity, exact path, audience, visibility, provenance, references, and generated metadata must agree. This is a candidate-only product procedure; it does not change governance, authority, approval, activation, or publication controls.

Before You Add a Page

Start with `content/public/engine-overview.md#start-here`, then read `sources/authority.json`, the sealed evidence-only map `artifacts/authority-seal/page-registry-synchronization-source-revision-map.json`, and `registries/sources.json`. Treat the authority file and its observed revision and digest as read-only. The map records evidence about the sealed inputs; it does not grant new authority. Compare every authority-listed source ID with its exact registry record, including locator, revision, digest, authority class, and `public-safe` disclosure. For this page the allowed IDs are exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`.

Stop before authoring if an authority input or required source is missing, stale, conflicting, unresolved, or not public-safe; if a digest or revision does not match the sealed observation; or if a claim cannot resolve in `registries/claims.json`. Keep that gap visible and escalate it through the governed route. Do not infer authority from an existing page, registry entry, `build/index.json`, generated prose, or an unregistered source, and do not copy private governance material into public content. Decide the intended relative `content/` path, stable page ID, exact H1 title, visibility, audiences, reading modes, claim references, and source references before writing. These facts are inputs to synchronization, not guesses made after a projection is generated.

Synchronize registries/pages.json

Write or review the semantic page and its `<!-- metadata -->` header first. The header-to-registry mapping is direct: `page_id` becomes registry `id`; the semantic file's repository-relative path becomes `path`; the first H1 becomes `title`; `visibility`, `audiences`, and `reading_modes` copy to the same-named registry fields; `claims` becomes `claim_refs`; and `source_refs` remains `source_refs`. Every claim and source reference must resolve in its registry, and a public page may name only the authority-listed records whose disclosure is `public-safe`.

For this page, preserve the stable identity `page.page-registry-synchronization` and the exact path `content/page-registry-synchronization.md`. Its registry object must keep the title `Page Registry Synchronization`, public visibility, audience `onboarding-teammate`, reading modes `executive`, `engineer`, and `ai`, an empty `claim_refs` list, and the three source IDs named above. Before saving, check both uniqueness dimensions: the ID occurs exactly once in the `pages` array and the exact path occurs exactly once. If either value already belongs to a different object, stop and resolve the identity collision; never invent a new ID from a filename and never append a second object for an existing ID or path. Update the existing matching object in the same authorized change only when the page metadata actually requires synchronization.

Review the diff as one bounded change. Confirm that the header `page_id` equals the registry `id`, the file path equals the registry `path`, the first H1 equals `title`, and the lists are equal in both value and intended order. Confirm that all references resolve and that no authority, source registry, journey manifest, generated file, publication pointer, or governing control was edited. A duplicate, unresolved reference, metadata disagreement, or public-safety mismatch is a stop condition, not a reason to weaken the check.

Verify the Change

From the repository root, run the deterministic projection command: `python3 tools/build_projection.py`. It derives `build/index.json` from the semantic and registry sources and is the approved writer for that generated file. Inspect the new entry for the same stable ID, exact path, H1 title, visibility, audiences, reading modes, resolved claim references, resolved source references, content revision, and authority identity. If the entry is wrong or absent, return to the semantic source or registry source; never hand-edit `build/` to make the projection pass. When a bounded comparison is needed, write a temporary output outside product sources and compare its bytes with the normal projection without treating that temporary file as authority.

Run the deterministic content check with `python3 tools/validate_content.py`. At the applicable candidate-verification stage, also run `python3 tools/release.py verify`; use the repository's defined scope and record the exact command, exit status, observed output, content revision, and relevant hashes for every command actually executed. Deterministic commands do not invoke an LLM. Preserve failed attempts beside later evidence, diagnose any non-reproducible projection instead of overwriting it, and do not describe an unrun check as passed. These checks validate product inputs and generated consistency; they do not attest to an independent route, validator authority, reviewer, approval, or terminal platform state.

Acceptance Checks

Accept the synchronization as a product candidate only when the evidence for the same page change is inspectable. First, the sealed authority revision and all three source records agree exactly on IDs, locators, revisions, digests, authority classes, and public-safe disclosure, with no unresolved stop condition. Second, the Markdown header and exactly one page-registry object agree on ID, exact path, title, visibility, audiences, reading modes, claim references, and source references; duplicate IDs and duplicate paths are absent. Third, the deterministic projection contains that same identity and resolved metadata, and generated bytes were inspected rather than hand-edited.

Fourth, the evidence records the exact projection, validation, and applicable release-verification commands, their actual statuses and results, and any reproducibility comparison or failure. Fifth, the handoff says only what the product evidence establishes. Candidate status does not mean approval, promotion, publication, scheduling, rollback, activation, a release pointer, or a live-directory write. Platform-owned attestations and independent reader or reviewer conclusions stay separate from product conclusions. If any authority, metadata, reference, projection, or command result is unresolved, stop with the gap visible and escalate it; do not guess, self-certify, or change the governing boundary.

Provenance: registry-listed (no candidate packet)
Registry Source: `content/page-registry-synchronization.md`