DOCUMENTATION PAGE

Publication Reader-Journey Gaps

Publication Reader-Journey Gaps

This operator guidance is for Lee and current company operators reviewing a parked candidate.

The publication-candidate packet and its readiness report leave several reader problems visible. A reader must find the authoritative source explanation, separate the selected candidate from a transition decision, and distinguish recorded product evidence from missing platform-owned attestations. The selected `repair-8ab0ca9952d8` record is candidate-only even though it is described as the strongest sealed candidate and as ready for human review. The report also leaves authority-snapshot drift, incomplete external serving boundary authority, artifact sufficiency, limited reader coverage, the environment-specific `Operation not permitted` observation, absent prior release and rollback facts, and the lack of a fresh independent review as documented gaps. None of those gaps may be filled with a guessed source, rollback target, webroot, command, approval, or serving outcome.

Closing a Documented Gap

Use the same closure record for every gap. First, name the gap and locate its authoritative explanation in `sources/authority.json`, `registries/sources.json`, the candidate packet, or the readiness report; record the exact path and the fact that is established or absent. Second, compare the current sealed source set and the immutable candidate records, keeping snapshot drift, missing external-boundary authority, limited reader or review scope, and null rollback facts explicit. Third, follow the applicable read-only command in the journey manifest and retain its exact command, exit status, and observed result beside the comparison. Close a gap only when the named evidence supplies the missing fact or an authoritative decision explicitly leaves it unresolved; otherwise route it to the owning operator or permitted evaluator. Updating this page or manifest can document a conclusion, but it cannot change sealed authority, grant approval, create a pointer, promote a candidate, publish content, or write a live directory.

For the source and snapshot gap, match every authority-listed ID to its public-safe registry record and record revision, locator, and digest; the current authority revision is `2026-08-08` and employee

modification is false. For evidence gaps, compare the candidate manifest, validation record, reader record, and review scope, preserving the one recorded environment blocker rather than calling it a recurring product defect. For serving and rollback gaps, require separately supplied external-boundary authority, artifact sufficiency, health-check evidence, an approved prior release, and a rollback target. If any prerequisite is absent, the repeatable closure result is unresolved and parked candidate evidence.

Candidate Packet Verification

Begin with the authoritative explanation, then inspect the packet and readiness records in this order: `sources/authority.json`, `registries/sources.json`, `content/publication-candidate-packet.md`, `releases/publication-readiness-report.md`, and `releases/publication-readiness-candidate.json`. Verify that the selected release is identified as `repair-8ab0ca9952d8`, that its status is `candidate` and its publication status is `candidate-only`, and that the product recommendation is scoped to human review. Compare the selected snapshot with the current authority without silently replacing one with the other. Confirm that the report's `previous_release` and `rollback_target` are null and that `rollback_available` is false; older candidates are comparison evidence, not rollback targets.

An independent reader may re-execute the manifest's commands from the repository root. `python3 -m json.tool sources/authority.json` checks that the authority record is readable; `python3 -m json.tool releases/publication-readiness-candidate.json` checks the selected-candidate facts; `python3 -m json.tool releases/repair-8ab0ca9952d8/release.json` checks the immutable release manifest; and `python3 -m json.tool journeys/user_journeys_manifest.json` checks the reader contract itself. These commands provide repeatable product-side observations only. Keep any preserved validator, reader, review, or platform evidence beside the new observation rather than replacing it, and record failures exactly.

Acceptance Checks

- **AC-1 — Find the authoritative explanation:** Starting at `sources/authority.json`, the reader matches all three listed source IDs to public-safe records in `registries/sources.json`, then locates the packet and readiness-report explanation for each identified gap without copying private governance material or inventing a source, claim, revision, or outcome.
- **AC-2 — Follow the closure procedure:** For each gap, the reader records the authoritative path, established or missing fact, owning boundary, exact allowlisted command, exit status, and observed

result; the reader compares candidate evidence and retains unresolved authority, serving, reader, review, and rollback facts instead of treating an absent fact as closed.

- **AC-3 — Verify candidate-only status:** The reader confirms the selected packet and release remain `candidate` / `candidate-only`, with no established approval, promotion, publication, schedule, rollback, release-pointer creation or change, or live-directory write, and keeps product conclusions separate from platform-owned attestations and any future transition.

Provenance: registry-listed (no candidate packet)
Registry Source: `content/publication-reader-journey-gaps.md`