PRINT EDITION

Documentation Portal

As-of Watermark: `20260816T190017Z`

Documentation Portal

Complete Print Edition

Watermark Cycle: `20260816T190017Z` | Total Pages: 44

Table of Contents

Active Needs-Lee Authority and Evidence Map

Audience: Lee, auto-orch operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Active Needs-Lee Authority and Evidence Map

Purpose

This Internal Knowledge map gives Lee and auto-orch operators one bounded view of the two active decisions in the `human-question record`. Its outcome is practical: a reader can choose the next smallest human decision from preserved evidence without treating product readiness, a reader result, or a candidate record as authority. This page is an advice-lane artifact, not an approval queue. It cannot approve a commissioning lineage, authorize a pilot, promote or publish a release, schedule work, change a pointer, roll anything back, or write a live directory.

Active Needs-Lee Map

The Open section names exactly **Human approval** and **Controlled pilot authorization**. The supplied reader manifest requires both journeys to trace to this page's AC-1 through AC-4 checks. The source record lists AC-DS-5 and AC-DS-7 in the operator-independent-boundary journey. Those labels are recorded as affected contract labels below; this page does not reinterpret them or claim that either label grants human or platform authority.

Human approval

- **Missing human authority:** Lee's explicit human-gate decision is missing: after reviewing the fresh commissioning evidence, Lee must decide whether the readiness contract supports the affected separately governed transition. A product-side pass or review recommendation cannot stand in for that decision.
- **Affected acceptance-contract labels: AC-DS-5** and **AC-DS-7**, both listed by the source record for the independent-boundary journey. The affected acceptance contract is the readiness-to-transition contract. The active question concerns the boundary between independent product evidence, candidate-only state, and the human decision; it does not assert either source-record label as satisfied.
- **Preserved evidence:** The `human-question record` identifies fresh run `c56754d00fd8` and candidate `ds002-c56754d00fd8` as the evidence awaiting the gate. The `result review` records a product-side pass with no findings, and `context` and `WHERE_AM_I` describe the retained candidate-only boundary. No failure artifact for this approval decision is available in the declared product inputs; none is invented here. These links are the available repository-relative decision evidence.
- **Smallest actionable Lee decision:** Review that fresh lineage and record only whether the readiness contract supports the affected next transition, with any resulting authority action left to its separately governed owner. Lee need not decide promotion, publication, scheduling, rollback, pointer mutation, or a live-directory write as part of this product map.
- **Reversible default:** Keep candidate `ds002-c56754d00fd8` parked and unchanged as candidate-only evidence. Leave any publication pointer absent and do not promote, publish, schedule, roll back, or write a live directory while the human-gate decision is open.

Controlled pilot authorization

- **Missing human authority:** Lee's explicit authorization for a controlled pilot is missing. The `human-question record` makes this a later decision, conditional on acceptance of a fresh commissioning lineage; it is not implied by approval review or by product readiness.
- **Affected acceptance-contract labels: AC-DS-5** and **AC-DS-7**, the two labels that the source record traces to the independent-boundary journey. The affected acceptance contract is the post-acceptance controlled-pilot transition contract. The labels identify the relevant evidence and transition boundary in the source record, not an authorization to run a pilot. The pilot decision remains Lee's authority.

- **Preserved evidence:** The `sprint plan`, `context snapshot`, `result review`, and `milestone record` preserve the commissioning lineage, product-side readiness conclusion, and candidate-only status with no publication pointer. No failure artifact for this pilot authorization is available in the declared product inputs; the available decision evidence is linked here instead of being replaced with an invented failure or approval.
- **Smallest actionable Lee decision:** Only after the fresh commissioning lineage has been accepted, decide whether to authorize or decline the narrow controlled pilot. This decision must remain separate from any later promotion, publication, scheduling, rollback, pointer change, or live- directory action, each of which needs its own authority boundary.
- **Reversible default:** No controlled pilot is authorized. Keep the candidate parked and unchanged, preserve its evidence, and leave promotion, publication, scheduling, rollback, pointer mutation, and live-directory writes untouched until Lee records the applicable decision.

Decision Queue

The queue is ordered by dependency, not by product confidence. First, Lee may review `c56754d00fd8` and `ds002-c56754d00fd8` and decide whether the readiness contract supports the affected human-gate transition. Second, and only after a fresh commissioning lineage is accepted, Lee may decide whether a controlled pilot is authorized. Auto-orch can prepare and preserve evidence for either review, but it cannot collapse the two decisions, infer consent from silence, or turn a product conclusion into approval. Until a decision is recorded, the smallest safe action is no transition: retain candidate-only state and do not promote, publish, schedule, roll back, create or change a pointer, or write a live directory.

Evidence and Authority Boundary

The sealed `authority record` says the employee may not modify the authority baseline, and the `source registry` contains the three public-safe sources used by this page. The repository evidence supports product conclusions about the fresh run, readiness review, lineage, and candidate-only status. The preserved validator evidence supplied for this step remains separate platform-owned gate evidence; it is not replaced by this page's prose or by a product command claim. No failure artifact for either active decision is present in the declared product inputs, so this map links the available decision evidence and records that absence explicitly.

Candidate-only means that evidence is retained behind the local publication boundary. It is not human approval, controlled-pilot authorization, promotion, publication, rollback readiness, scheduling authority, pointer creation, or a live-directory result. The governing repository and platform owners retain authority for those transitions and for route, identity, validator, terminal, and approval attestations. This page therefore recommends a reversible parked state until Lee decides; it does not perform or imply any external transition.

Acceptance Checks

- **AC-1 — Coverage:** Every active needs-Lee item in the Open section of `questions-for-the-human.md`, namely Human approval and Controlled pilot authorization, has one map entry with its own evidence and decision fields.
- **AC-2 — Authority and contract:** Each map entry names the missing human authority and affected contract—readiness-to-transition for Human approval or post-acceptance pilot transition for Controlled pilot authorization—and records the supported AC-DS-5 and/or AC-DS-7 labels without treating them as Lee's authority.
- **AC-3 — Evidence:** Each map entry links preserved repository-relative decision evidence or explicitly records that no failure artifact is available in the declared product inputs; neither entry invents a failed run, approval, platform attestation, or transition result.
- **AC-4 — Decision and default:** Each map entry gives Lee the smallest actionable decision and states a reversible default that keeps the candidate parked and unchanged without promotion, publication, scheduling, rollback, pointer mutation, or live-directory writing.

Authoritative Reader-Journey Contract Compatibility Checks

Audience: agent-orch maintainers, governed-run operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Authoritative Reader-Journey Contract Compatibility Checks

Purpose

This page gives agent-orch maintainers and governed-run operators a repeatable compatibility check for a reader-journey manifest, its page metadata, and its evidence claims. It keeps the result candidate-only: the procedure can show a product mismatch or a reproducible product pass, but it cannot change sealed authority, grant validator authority, approve a run, publish content, or create a release pointer. Read the exact manifest names and goals as the contract a later independent reader must be able to replay.

Authoritative Contract

The sealed authority is the source boundary. `sources/authority.json` has revision `2026-08-08`, names exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and sets `employee_may_modify` to `false`; the matching records in `registries/sources.json` must remain public-safe. The compatibility manifest is authoritative for its own enumerated AC-1 through AC-4 criteria, exact journey `name` strings, natural-language `goal` text, `authority` values, `command_allowlist` strings, and `traces_to` mappings. A journey authority is permitted only when it is exactly `human`, `mission`, `author`, or `exploratory`; a required non-exploratory journey uses `human`, `mission`, or `author`, and an exploratory journey cannot replace required coverage.

Exact journey names are stable evidence keys, not display suggestions. A `commands_run` claim records the exact allowlisted command, exit status, and observed result for the manifest journey whose name it carries. The natural- language goal explains what an independent reader starts with and can observe; it does not authorize extra commands. A command claim is valid only when its string matches one `command_allowlist` entry byte-for-byte. Product evidence from that claim remains separate from platform-owned attestations.

Compatibility Checks

Run this procedure from the repository root before accepting a compatibility result:

1. Open `journeys/reader-journey-contract-compatibility.json` and enumerate AC-1, AC-2, AC-3, and AC-4. For every required non-exploratory journey, compare `traces_to` with the declared AC IDs and compare the full `name` and natural-language `goal` with the journey being evaluated. If any AC has no trace, or a journey traces to an undeclared AC, record a missing-AC-trace mismatch and stop; do not invent a trace or silently drop the journey.
2. Validate each journey `authority` as an exact enum string. A value such as `Mission`, `operator`, `human`, or a missing value is malformed even if a reader could guess its intent. Required non-exploratory coverage must not be converted to exploratory coverage to hide the malformed value.
3. For each `commands_run` claim, compare the command string byte-for-byte with the manifest `command_allowlist`. A command that is merely similar, reordered, or wrapped in an unlisted shell command is not permitted. Record the offending command, the journey name, and the allowlist comparison as a product mismatch; do not execute an unpermitted command to obtain a pass.
4. Key every result by the exact manifest journey `name`, including case, punctuation, and spacing. A failed result keyed by a journey ID, filename, shortened label, translated title, or any other string is not compatible; preserve it as a failed or mis-keyed record and request repair before interpreting its exit status or output.

The four checks are independent gates: a passing command cannot compensate for a missing AC trace, malformed authority, or mis-keyed result. Keep the exact failure, command, exit status, observed output, and comparison path available to the next reviewer so the procedure distinguishes a product conclusion from an unavailable platform execution.

Repair and Re-execution

Repair only the bounded product mismatch named by the comparison. Page prose belongs in `content/reader-journey-contract-compatibility.md`, synchronized page identity and metadata belong in its one object in `registries/pages.json`, and manifest names, goals, authorities, traces, or command strings belong in `journeys/reader-journey-contract-compatibility.json`. Do not modify `sources/authority.json` or `registries/sources.json` to make a reference pass, and do not hand-edit generated output, releases, publication pointers, or a live directory. A repaired field must still be checked against the sealed source boundary and the page header/registry pair.

After repair, an independent reader reopens the manifest, repeats the four compatibility checks, and re-executes only an exact allowlisted command from the journey under review. The resulting `commands_run` claim must carry the same exact journey name, command string, exit status, and observed result. Retain the earlier failed or mis-keyed record beside the later result rather than overwriting history. A successful local product command establishes only the observed product result; it does not attest route selection, validator authority, approval, activation, promotion, publication, scheduling, rollback, or a release pointer.

Acceptance Checks

- **AC-1 — sealed source boundary:** The reader verifies the authority revision, exact three authority-listed source IDs, and `employee_may_modify: false`, then confirms that each matching registry record is public-safe without copying private governance material or editing the protected source files.
- **AC-2 — journey compatibility:** The reader enumerates the exact manifest journey names, confirms each natural-language goal and required `traces_to` mapping covers one or more of AC-1 through AC-4, and rejects a missing trace, duplicate identity, malformed authority enum, or exploratory substitute for required non-exploratory coverage.
- **AC-3 — command evidence:** The reader compares every `commands_run` command claim byte-for-byte with `command_allowlist`, including `python3 -B tools/validate_content.py`, and rejects a command not permitted by that list while retaining its exact exit status and observed result as a product-side compatibility finding.
- **AC-4 — independent keyed re-execution:** An independent reader re-runs an exact allowlisted command and records the result under the exact manifest journey name. A failed result keyed by anything else is incompatible, and a later pass does not erase the earlier failure or expand the candidate-only boundary.

Workstream Selection

This page is the **Internal Knowledge** workstream for agent-orch maintainers and governed-run operators. The sealed selection record makes Internal Knowledge the reproducible choice because it is eligible, underserved in scheduling tier 1, and oldest among the eligible underserved queues before any item-level impact or feasibility score is considered; its selected candidate is the Failure-Triage Decision Guide. **Employee Onboarding** was not selected because it is eligible but not marked underserved, so the ordered rule excludes it before item-level scoring. **Customer Education** was likewise eligible but not underserved and was excluded at the same stage. Both remain visible in the comparison and are not permanently excluded. This workstream choice scopes the documentation route only; it does not schedule work or authorize a transition.

Blocker Decision Brief

Audience: owner, onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Blocker Decision Brief

Blocked Repair

The strongest unresolved repair is candidate `repair-8ab0ca9952d8`, selected in `releases/publication-readiness-candidate.json` and described by `releases/publication-readiness-report.md`. Its product-side evidence reports passed reader evidence with no findings, a prior sealed review with no findings, and `promotion_ready: true`, while its release manifest remains `status: candidate` and `publication_status: candidate-only`. The blocked transition is the separately governed move beyond candidate-only status. The concrete blocker is that human approval is still open; readiness evidence is not approval and this brief does not authorize a repair, pilot, promotion, or publication.

Evidence

The decision packet is `questions-for-the-human.md`, `releases/publication-readiness-candidate.json`, `releases/repair-8ab0ca9952d8/release.json`, its `validation.json`, `evidence/reader.json`, and `evidence/review.json`. The human-question record says approval must be reviewed at the human gate and separately identifies controlled-pilot authorization as later authority. The selected candidate records explicit operator approval and evidence as its promotion policy; the reader recommendation is scoped to a product conclusion, not authorization. The preserved system-validator record for this step reports one passed `command_succeeds` result with exit status 0 and evidence hash `6199ece52622548300b9375f68b4ff4aa51e5fd9921ebe6a09532cd106eceb44`; it is authoritative evidence and was not rerun here.

Authority Needed

Lee needs to review the sealed commissioning lineage and make the explicit human-gate decision on whether the affected, separately governed transition may proceed. That is the smallest required decision. The decision does not grant this repository authority to change `sources/authority.json`, routing, validator authority, activation rules, publication policy, or the governing repository. `sources/authority.json` records `employee_may_modify: false`, and the product evidence cannot supply platform-owned route, identity, validator, execution, evidence-chain, terminal, or approval attestations. If those attestations are required by the governing gate, they must be obtained from their platform owner rather than inferred from product readiness.

Reversible Default

Until Lee records that decision, leave `repair-8ab0ca9952d8` parked and unchanged as a candidate. Keep `publication/current-release.json` unchanged; do not promote, publish, schedule, re-arm, or attempt rollback. This is reversible because the candidate packet and its linked evidence remain available for later review. The selected release records `previous_release: null`, `rollback_target: null`, and `rollback_available: false`, so older candidates must not be invented as rollback targets. This brief performs no transition and does not ask for any product-side mutation beyond retaining the current state.

Recommendation

Recommend that Lee review `repair-8ab0ca9952d8` at the human gate and record the smallest explicit decision: allow the separately governed transition to proceed, or keep the candidate parked. The recommendation is supported by the newest sealed candidate's two-page coverage, passed product reader evidence with no findings, prior sealed review evidence with no findings, and explicit candidate-only facts. “Ready for controlled pilot” is only a product-side recommendation in the reader record; it is not Lee's approval, controlled-pilot authorization, platform attestation, promotion permission, or publication instruction. Missing authority remains a blocker, not a reason to infer consent.

Publication Boundary

The candidate remains behind the local publication boundary `publication/current-release.json`. The selected release and publication readiness records state candidate-only status, no pointer creation, no promotion, no publication, no scheduling, no rollback, and no live-directory write. This authoring step changes only this decision brief and does not alter the candidate, release pointers, generated projections, tests, source authority, governance, routing, or a live directory. Any later transition is outside this record and requires the applicable human and platform authority; candidate readiness must not be read as permission to cross that boundary.

Acceptance Checks

- AC-1 — The reader can identify candidate `repair-8ab0ca9952d8` as the blocked repair and can trace the missing human approval and supporting candidate, validation, reader, review, and human-question evidence.
- AC-2 — The reader can state that Lee's explicit human-gate decision is required for the separately governed transition, while distinguishing that authority from product conclusions and platform-owned attestations.
- AC-3 — The reader can distinguish the reversible default of keeping the candidate parked and unchanged from the recommendation that Lee review it and record an explicit decision; no transition is performed by this brief.
- AC-4 — The reader can confirm the candidate-only publication boundary, including that no release pointer, promotion, publication, scheduling, rollback, or live-directory write occurred, without inferring later permission.

Candidate Evidence Contract Decision Packet

Audience: Lee, agent-orch contract owner | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Candidate Evidence Contract Decision Packet

Purpose

This Internal Knowledge packet gives Lee, the agent-orch contract owner, a bounded decision record for a declaration conflict. The human-readable candidate-evidence contract currently declares five checks and a JSON-tool allowlist, while the executable, model, and valid-fixture route currently declares three pinned journeys with exact `-B` pytest or smoke commands. The packet records those facts without selecting a canonical contract, changing an input, or treating any declaration as proof that a command ran.

The intended outcome is a human decision about which route, if any, should later be reconciled and who should own that migration, while every input remains unchanged. The first declaration is the human-readable five-check contract in docs/candidate-evidence-preflight-contract.md, whose companion route is described as JSON-tool based. The second is the three-journey executable, model, and fixture route in the candidate-evidence handoff manifest and model. These are current expectations only: no declaration, model, fixture, or allowlist entry is execution proof or approval.

The deterministic preflight check exposes the practical form of this conflict: the current `journeys/user_journeys_manifest.json` is the separate publication-reader-gap manifest, not the pinned three-journey candidate- handoff manifest expected by the model and fixtures. This packet's reader manifest now preserves that pinned three-journey shape, including its evidence schema, review-verdict requirement, exact allowlist, and expected results, while recording the user-manifest mismatch as unresolved. That observation is a declaration-reconciliation defect in this route, not a product runtime defect. The separate reader-gate observation — journey `journey.resolve-reader-journey-harness-decision`, command `python3 tests/check_documentation_smoke.py`, exit status `1`, and output `Operation not permitted` after localhost socket access was denied — remains an environment-specific evaluation blocker and supplies no recurrence, affected claim, or repair target.

Workstream Selection

The selected workstream is **Internal Knowledge**, as declared by journeys/candidate-evidence-handoff-manifest.json. The audience is **Lee, the agent-orch contract owner**, and the intended outcome is a bounded, operator-facing evidence handoff that records acceptance coverage, artifact authority, source references, validation commands, output locations, and unresolved blockers without crossing a platform-owned transition boundary.

Employee Onboarding and **Customer Education** were not selected for this bounded cycle because the sealed candidate-evidence manifest names Internal Knowledge as the workstream and supplies no competing audience, intended outcome, or authority declaration for either alternative. This is a scope boundary, not a product judgment about those workstreams; selecting one would expand the packet beyond the declared evidence question and require new authority rather than resolving the present contract conflict.

Conflicting Declarations

The observed human-readable declaration is `docs/candidate-evidence-preflight-contract.md`, with its stated companion `journeys/user_journeys_manifest.json`: the contract names AC-1 through AC-5, five canonical natural-language journeys — `As a maintainer, verify complete candidate-evidence contract coverage.`, `As an independent reader, match each journey name to its contract declaration.`, `As a maintainer, check every preflight command against the allowlist.`, `As a reviewer, validate the evidence record schema before accepting a result.`, and `As an operator, require the independent review verdict before handoff completion.` — and the JSON-tool commands `python3 -m json.tool journeys/user_journeys_manifest.json` and

`python3 tools/validate_content.py`. The separate executable/model/fixture route in `journeys/candidate-evidence-handoff-manifest.json` and the repaired reader manifest `journeys/candidate-evidence-contract-decision-packet.json`, `models/candidate_evidence_handoff_contract.json`, and the candidate-evidence fixtures pins exactly three journeys — `journey.candidate-evidence-handoff-preflight`, `journey.candidate-evidence-handoff-template`, and `journey.candidate-evidence-handoff-documentation-validation` — with AC-1 through AC-3 and the allowlist `python3 -B tests/check_documentation_smoke.py` plus `python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py`. The current user-journey file instead identifies `journey-manifest.publication-reader-journey-gaps` and carries the three journeys `journey.find-authoritative-publication-gap-explanation`, `journey.follow-documentation-gap-closure-procedure`, and `journey.verify-candidate-only-packet-boundary`, with four JSON-tool commands. The packet manifest is synchronized to the executable/model/fixture route for its reader-facing evidence shape; the existing test oracle still treats the separate user-manifest path as its input, so this remains an observed route inconsistency to reconcile at the owning boundary, not proof that either declaration wins. The handoff manifest adds the selected workstream, goals, audience, entry points, and journey-level traces; the model and packet manifest add the read-only evidence schema, preflight tool, review-verdict requirement, and expected journey exit codes; neither addition canonizes the five-check document. The valid fixture mirrors the three journey names, traces, exact commands, and expected exit codes `[0, 0, 1]`. The disallowed-command fixture intentionally presents `python3 tools/validate_content.py` as a command claim for rejection because it is not an exact member of the executable/model allowlist. These manifest, model, fixture, and allowlist contents are preserved declarations, not proof that a command ran or that the route was approved. The current declarations remain separate and unresolved; no input or owner is claimed to have changed or approved them.

Decision Matrix

Area	Declaration currently observed	Owner or owner-to-confirm	Decision Lee must make	Recommended reversible default
Authority	`sources/authority.json` records revision `2026-08-08`, exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, with `employee_may_modify: false`; the source registry marks these public-safe.	The separate governing repository and the named source authorities; Lee is not authorized to edit this boundary.	Confirm that authority is a boundary for this packet, not a tie-breaker that canonizes a product declaration.	Preserve authority and `registries/sources.json` unchanged; record any authority question for its owner.
Manifest	The contract names five checks and a JSON-tool route, while the current `journeys/user_journeys_manifest.json` identifies the publication-reader-gap manifest; `journeys/candidate-evidence-handoff-manifest.json` and this packet's reader manifest carry the three candidate-handoff journeys and exact `-B` route expected by the model and fixtures.	Contract/manifest owner-to-confirm; the current files provide declarations, not an approved migration order.	Decide which competing declaration set a later governed update should reconcile, without treating this packet's synchronized reader shape as canonical authority.	Preserve the competing input manifests, keep this packet manifest aligned to the pinned handoff contract, record the deterministic mismatch, and keep the conflict unresolved.
Model	`models/candidate_evidence_handoff_contract.json` declares a read-only model with three journeys, AC-1 through AC-3, the pytest/smoke allowlist, and a review-verdict path.	Model owner-to-confirm with the agent-orch contract owner.	Decide whether the model is intended to describe the same contract as the five-check human-readable document.	Keep the model unchanged and do not promote it to canonical authority by inference.
Fixture	`valid.json` carries the three pinned journey records, exact commands, and expected exit codes `[0, 0, 1]`; `disallowed-command.json` deliberately carries the non-member `python3 tools/validate_content.py` claim that the executable route must reject.	Test and fixture owner-to-confirm; fixture behavior is evidence of current expectations, not approval.	Decide whether fixtures should migrate only after a canonical contract is approved.	Leave both fixtures unchanged pending an approved later update; retain valid and negative expectations.
Allowlist	The human-readable contract lists the exact JSON-tool commands `python3 -m json.tool journeys/user_journeys_manifest.json` and `python3 tools/validate_content.py`; the executable/model/fixture route lists exact `-B` pytest and smoke commands, with whitespace and command identity significant.	Contract and manifest owner-to-confirm; validator maintainers implement only an approved declaration.	Decide which allowlist, if any, belongs in a later canonical contract update.	Leave both allowlists unchanged pending an approved later update; defer canonicalization.

Area	Declaration currently observed	Owner or owner-to-confirm	Decision Lee must make	Recommended reversible default
Migration ownership	No approved migration plan, owner assignment, or compatibility window is supplied for reconciling the five-check and three-journey routes.	Owner-to-confirm by Lee and the agent-orch contract owner through the governed contract route.	Name the accountable owner and approval path before any synchronized edit is proposed.	Make no migration edits; preserve the conflict and request an explicit later contract update.

The matrix is a decision aid, not a change record. Its owner entries distinguish what the sealed inputs establish from ownership that still requires confirmation. The one recommended default is reversible: leave every competing input unchanged pending an approved later update, while keeping this packet's reader-facing manifest synchronized to the already pinned executable/model/ fixture shape. This preserves the competing declarations and keeps a successful check on one route from being silently reused as evidence for the other. The current preflight-suite mismatch is therefore a reason to route a source-boundary reconciliation, not permission to edit the user-journey manifest, model, fixtures, validator, or competing allowlists in this packet.

Recommended Reversible Default

Keep the five-check/JSON-tool declaration and the three-journey/ `-B` pytest-or-smoke declaration unchanged, label the relationship as an unresolved contract conflict with a recorded deterministic mismatch, and park this packet for Lee's decision. The packet reader manifest is synchronized to the pinned three-journey evidence shape so its schema and command claims are reader-verifiable, but that output does not edit the human-readable contract, user-journey manifest, candidate-handoff manifest, model, validator constants, fixtures, or competing allowlists. A later approved contract update may define a canonical route and a migration owner; until then, no declaration wins, no expected exit code becomes execution evidence, and no missing owner is filled in by inference.

Approval Questions

Lee should answer only the bounded contract questions: Which declaration set is intended to be canonical for this candidate-evidence route? Who owns the reconciliation and the migration plan? Should the five human-readable checks, the three executable/model/fixture journeys, or an explicitly revised set be carried forward? What evidence and approval are required before synchronizing the competing contract, manifests, model, fixtures, and allowlist? Until those answers are supplied through the governed route, this packet records a conflict rather than an approval. Lee must also decide whether the observed suite mismatch should be routed to the owning manifest/model boundary, while keeping the separate localhost-denied reader observation as an environment-specific blocker. No manifest, model, fixture, allowlist, or owner is currently claimed to be changed or approved, and none of these questions authorizes execution, publication, promotion, scheduling, rollback, or another transition.

Candidate-Only Boundary

This is candidate-only Internal Knowledge material for Lee's decision. It does not approve, promote, publish, schedule, activate, roll back, create or change a release pointer, or write a live directory. It also does not establish a product runtime defect, affected claim, independent review verdict, or platform-owned attestation. The declaration-reconciliation mismatch is the bounded route issue recorded here; its owning repair target remains unset until Lee's governed decision. The only safe next state is the reversible parked candidate with all conflicting declarations preserved. A later contract owner may reconcile them through an approved update; that future action is outside this packet and must not be inferred from its page, registry entry, or journey manifest.

Acceptance Checks

- **AC-1:** The reader compares the human-readable five-check and JSON-tool declaration, the current publication-reader-gap user manifest, and the executable/model/fixture route's three pinned journeys and exact `-B` pytest or smoke commands, preserving the deterministic mismatch and each declaration as unresolved. The packet reader manifest exposes the same evidence schema and execution declarations without canonizing either route.

- **AC-2:** The reader selects the recommended reversible default: preserve the human-readable contract and competing input manifests, model, fixtures, and allowlists unchanged; keep this packet's reader manifest synchronized to the pinned handoff shape; route declaration reconciliation to its owner; and defer canonicalization to an approved later contract update.
- **AC-3:** The reader preserves the candidate-only approval boundary, treats the localhost-denied reader result as environment-specific, and does not infer approval, promotion, publication, scheduling, activation, rollback, release-pointer change, live-directory work, product runtime defect, affected claim, or platform-owned attestation from this packet.

Candidate Evidence Handoff Preflight

Audience: agent-orch maintainers preparing an operator-facing evidence handoff, Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Candidate Evidence Handoff Preflight

This page is a bounded troubleshooting guide for auto-orch operators and maintainers preparing a candidate-only evidence handoff. It keeps product conclusions, source authority, reader evidence, and platform-owned attestations distinct. A failed observation is still evidence: preserve its exact command, exit code, output, and owner rather than rewriting it as success.

Preflight Failure Triage

Start with the complete read-only result, not with a guessed repair. Preserve the evidence path, every error string, the first observable failure, and the declared manifest and contract paths. `run_preflight()` in `tools/candidate_evidence_preflight.py` reads those inputs and the independent review verdict; it does not execute a claimed command or turn a declaration into proof. Use `docs/candidate-evidence-preflight-contract.md`, the executable validator, `sources/authority.json`, and `registries/sources.json` as the source-linked diagnosis boundary. Classify the result as schema/evidence shape, command allowlist, journey identity/trace, or fixture/review-path failure, then repair only supplied candidate evidence or route a declaration problem to its owner.

The deterministic focused check is `python3 -m pytest -p no:cacheprovider tests/test_candidate_evidence_preflight.py`; compare the reported case with the matching fixture in `tests/fixtures/candidate_evidence/`. A known declaration-drift symptom is a contract that describes five checks and JSON-tool commands while the executable validator, model, and candidate fixtures pin three journeys and the exact `-B` pytest/smoke commands. That conflict is itself evidence: do not choose a side or hide it in this page. If an observation cannot be tied to an authoritative source, exact journey, and reproducible command, retain it as `unresolved` and stop rather than borrowing a nearby result.

Schema and Evidence Shape Failures

Observable symptom: The preflight reports `schema_version`, `kind`, `journeys`, a missing required field, `observed_result`, an invalid status, or coverage errors. A common example is `tests/fixtures/candidate_evidence/malformed-evidence.json`, whose first command claim has an exit code but no observed-result field.

Authoritative diagnosis: Read the required top-level, journey, and command claim fields in `docs/candidate-evidence-preflight-contract.md`, then confirm the executable checks in `validate_evidence()` and the schema constants in `tools/candidate_evidence_preflight.py` (also mirrored by `models/candidate_evidence_handoff_contract.json`). The record needs schema version 1, kind `candidate-evidence-preflight-evidence`, all declared journeys, exact names, non-empty `steps_taken`, `source_refs`, and `traces_to`, a string-list `blockers`, and command claims with an integer `exit_code` and non-empty `observed_result`. The validator requires each evidence record's status to be `passed`; that says the record is structurally complete for preflight, not that every claimed command exited zero. The pinned smoke claim therefore preserves its expected exit code 1 and its failed observation.

Safe remediation: Rebuild the record from the actual run and add only observed values, retaining failures in `observed_result` and `blockers`. Populate source references from the declared artifact, not from memory. Do not loosen the schema, delete a failed journey, replace an empty field with plausible prose, or change the validator to accept malformed evidence.

Deterministic confirmation: Run `python3 -m pytest -p no:cacheprovider tests/test_candidate_evidence_preflight.py::CandidateEvidencePreflightTests::test_malformed_evidence_is_rejected` against `tests/fixtures/candidate_evidence/malformed-evidence.json`, whose first command claim omits `observed_result`; use `tests/fixtures/candidate_evidence/valid.json` for the complete shape. If the real record still lacks an authoritative observation, stop with that field unresolved and escalate to the evidence owner; do not fill it with a summary of what the command was expected to do.

Command-Allowlist Failures

Observable symptom: The error includes `command allowlist`, or a command looks almost right but differs by a leading/trailing space, doubled space, newline, or a different executable. `tests/fixtures/candidate_evidence/disallowed-command.json` deliberately claims `python3 tools/validate_content.py` for a journey that is pinned to a different command.

Authoritative diagnosis: The command allowlist and each journey's exact command are pinned by `tools/candidate_evidence_preflight.py`, the candidate model, and the focused tests. For the executable candidate-handoff route, the only two exact allowlist strings are `python3 -B tests/check_documentation_smoke.py` and `python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py`. The first two pinned journeys claim the pytest command with exit code 0; the documentation-validation journey claims the smoke command with exit code 1. The test `test_command_claims_are_not_normalized_before_allowlist_matching` confirms that whitespace is not silently normalized, and a claim must also equal its journey's declared command. The human-readable contract currently contains a different JSON-tool allowlist; that declaration conflict must be escalated, not papered over by this guide.

Safe remediation: Copy the exact allowlisted string into the matching journey claim and preserve the command's separately observed exit code and output. If the needed check requires another route, do not add it to the candidate record or alter the allowlist as a page repair; ask the owner of the manifest/contract for a governed change. A command being useful or successful does not make it evidence for a different journey.

Deterministic confirmation: Run `python3 -m pytest -p no:cacheprovider tests/test_candidate_evidence_preflight.py::CandidateEvidencePreflightTests::test_disallowed_command_is_rejected tests/test_candidate_evidence_preflight.py::CandidateEvidencePreflightTests::test_command_claims_are_not_normalized_bef` against `tests/fixtures/candidate_evidence/disallowed-command.json`. If the exact command cannot be run in the permitted route, preserve the failure and stop; never substitute a locally convenient command, strip whitespace for a claim, or report that the allowlist was satisfied.

Journey Identity and Trace Failures

Observable symptom: The result reports a journey `.name`, duplicate or missing `journey_id`, journey coverage, `traces_to`, acceptance mapping, or pinned handoff mismatch. `tests/fixtures/candidate_evidence/invalid-journey-name.json` shows the typical case: the ID remains present while the name silently drops required wording.

Authoritative diagnosis: The pinned IDs, names, position-specific traces, commands, and expected exit codes are the constants in `tools/candidate_evidence_preflight.py`. For this handoff they cover exactly AC-1, AC-2, and AC-3, in order, with traces `["AC-1"]`, `["AC-2"]`, and `["AC-3"]`. The expected IDs are `journey.candidate-evidence-handoff-preflight`, `journey.candidate-evidence-handoff-template`, and `journey.candidate-evidence-handoff-documentation-validation`; their names must be copied byte-for-byte from the pinned declarations and fixtures. The contract's exact-name and coverage rules explain why an identifier, shortened name, reordered trace, or extra journey is not equivalent. Compare `journeys/user_journeys_manifest.json` with the pinned candidate manifest, model, tool, and fixture; do not silently reconcile a mismatch by editing this page.

Safe remediation: Recreate the evidence entry from the current pinned manifest, copying the journey ID and name byte-for-byte and retaining the declared `traces_to` value. Keep a failed observation attached to its original journey; do not move it to a journey that happens to pass. If the manifest, contract, and executable expectations disagree, preserve all three facts and escalate the declaration conflict rather than changing a registry or source reference in this page.

Deterministic confirmation: Run `python3 -m pytest -p no:cacheprovider tests/test_candidate_evidence_preflight.py::CandidateEvidencePreflightTests::test_invalid_journey_name_is_rejected tests/test_candidate_evidence_preflight.py::CandidateEvidencePreflightTests::test_journey_acceptance_mapping_is_pinned_` against `tests/fixtures/candidate_evidence/invalid-journey-name.json`. If the ID, exact name, or trace cannot be established from synchronized declarations, stop checklist completion and escalate to the manifest/contract owner. A successful check on the wrong journey is not coverage, and a current publication-reader manifest is not evidence for this candidate handoff.

Fixture and Review-Path Failures

Observable symptom: A fixture is missing required evidence, the evidence does not name `code-reviews/content-review.verdict.json`, the verdict file is absent or unreadable, or its JSON has the wrong schema, verdict, or forbidden finding severity. `missing-review-path.json` tests the evidence-side

path omission; `run_preflight()` separately reads the required file from the supplied repository root.

Authoritative diagnosis: The review-verdict requirement and its exact relative path are defined in `docs/candidate-evidence-preflight-contract.md`. `validate_review_verdict()` in `tools/candidate_evidence_preflight.py` requires schema version 1, verdict `pass` or `clean`, an array of findings, and no High or Critical finding. `run_preflight()` resolves the required path under the supplied `--root` and reads it independently. The valid fixture is a shape example, not proof that the independent artifact exists in the current run; a named path is not the same as a readable verdict. The `missing-review-path.json` fixture tests the evidence-side path omission, while the CLI path check tests a separately supplied verdict file.

Safe remediation: Use the valid fixture to check field shape, then supply the independently produced verdict at the exact path under the preflight root. Keep a missing, stale, malformed, or unverified verdict as a blocker; do not fabricate a verdict, copy one from another handoff, or treat a local content validator result as independent review. Preserve any failed fixture or review read as evidence for the owner who can provide the artifact.

Deterministic confirmation: Run `python3 -m pytest -p no:cacheprovider tests/test_candidate_evidence_preflight.py::CandidateEvidencePreflightTests::test_missing_review_path_is_rejected tests/test_candidate_evidence_preflight.py::CandidateEvidencePreflightTests::test_run_preflight_requires_a_readable_inc` and compare the record with `tests/fixtures/candidate_evidence/valid.json`. For a controlled CLI replay, use the test's read-only shape: `python3 tools/candidate_evidence_preflight.py --evidence <candidate-evidence.json> --manifest journeys/user_journeys_manifest.json --contract docs/candidate-evidence-preflight-contract.md --root <preflight-root> --json`. If the exact review artifact is unavailable, malformed, or not independent, or fixture and manifest declarations diverge, stop and escalate; a preflight cannot self-certify independent review.

Escalation Boundary

Stop and escalate when an authority, source reference, exact journey identity, allowlisted command, observed result, fixture, or independent review artifact is missing or contradictory; when a command needs a route outside the pinned allowlist; or when a failed observation could be interpreted as a product defect without authoritative recurrence and impact evidence. Failed observations remain evidence and must be carried forward verbatim with their owner and blocker. A disagreement between the human-readable contract, manifest, model, executable constants, or focused fixtures is a declaration failure, not permission to edit whichever file makes the preflight pass. This page may describe a boundary, but it cannot invent provenance, retry counts, affected claims, platform attestations, approval, or a repair target.

The handoff remains candidate-only. Neither this guide nor a preflight pass approves, promotes, publishes, schedules, activates, rolls back, changes a release pointer, or writes a live directory. Those transitions and platform-owned attestations belong to their governing authorities. When in doubt, retain the candidate, record the deterministic failure, and route the unresolved question to the authority named by `sources/authority.json` and `registries/sources.json` rather than making the evidence appear complete.

Candidate Evidence Handoff Preflight and Template

Audience: agent-orch maintainers preparing an operator-facing evidence handoff, future Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Candidate Evidence Handoff Preflight and Template

Purpose

Workstream: Internal Knowledge.

Audience: Lee and auto-orch operators, including Agent-Orch maintainers preparing an operator-facing evidence handoff, together with future Internal Knowledge playbook authors and independent readers who need to reconstruct the handoff.

Intended outcome: A bounded, auditable packet that lets a maintainer show what was checked, which authority owns each fact, where the result lives, and what remains unresolved without turning a product conclusion into a platform attestation or a transition decision.

Success hypothesis: If acceptance coverage, smoke artifacts, command allowlisting, and output locations are checked before the checklist is drafted, a reader can re-execute the documented validation and distinguish a missing fact from a failed product check.

Validation journey: `journey.candidate-evidence-handoff-documentation-validation` in `journeys/candidate-evidence-handoff-manifest.json`.

The handoff is candidate-only documentation evidence. It does not modify the sealed authority, copy private governance material, establish a recurring product defect from one environment-specific reader failure, or authorize approval, activation, promotion, publication, scheduling, rollback, a release pointer, or a live-directory write. Keep those boundaries visible while the operator assembles the evidence.

Preflight Procedure

Before drafting a checklist, read the sealed preflight receipt at `artifacts/preflight/candidate-evidence-handoff.json` and confirm that the selected workstream is **Internal Knowledge**, the audience is the maintainers and operators named above, and the intended handoff outcome is still bounded. Then verify acceptance coverage: AC-1, AC-2, and AC-3 must each have a concrete artifact, a source reference, an authority owner, and at least one journey trace. A journey that is exploratory may inform discovery, but every non-exploratory journey must trace to one or more of those three checks.

Next inspect the smoke artifacts before writing prose about execution. Confirm that `tests/smoke_manifest.agent-orch.json` names the expected start and check contract, that `tests/check_documentation_smoke.py` resolves its page and registry inputs, and that the documentation-validation journey allowlists the exact re-executable command `python3 -B tests/check_documentation_smoke.py`. The command is a reproducible claim about how to evaluate the documentation; its presence in the manifest is not proof that a platform ran it. Record the observed exit status and output separately, including the current environment-specific `Operation not permitted` reader-gate blocker when that is the result. Do not replace it with a guessed product defect or a successful result from another route.

Finally, verify every command before checklist drafting: it must be a non-empty JSON string in `command_allowlist`, point to the declared repository inputs, and have a named output location. Compare the page metadata with its single entry in `registries/pages.json`, and compare the three authority-listed source IDs with their public-safe records in `registries/sources.json`. Preserve the independent validator evidence supplied by the governing handoff; do not self-certify it or rerun a preserved system-validator result as a substitute. The manifest's `python3 -B` commands are the reader-reexecution allowlist. A preserved record's declared command is copied verbatim as evidence and is not silently changed to the current reader command. Only after

acceptance coverage, smoke artifacts, and re-executable commands are all accounted for should the maintainer draft the checklist and mark any result as passed, blocked, or unresolved.

Evidence-Handoff Template

Copy one row for every required artifact and complete every column. “Authority” names the boundary that owns the fact, while “Source reference” identifies the repository-relative record or preserved evidence that a reader can inspect. “Validation result” must state what was actually observed, not what the operator expected. “Command” must be the exact allowlisted invocation used or claimed, and “Output location” must point to the resulting record. If a check was not run, say so. If the evidence is blocked, retain the exact blocker and leave the conclusion unresolved.

Required artifact	Authority	Source reference	Validation result
Sealed authority and source boundary	Platform-owned authority boundary	`sources/authority.json`; `registries/sources.json`	Preflight observed revision `2026-08-08`, exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, `employee_may_modify: false`, and public-safe records; AC-1 preflight passed
Selected workstream and handoff scope	Mission	`registries/workstream-selection-receipt.json`; `artifacts/preflight/candidate-evidence-handoff.json`	Receipt records **Internal Knowledge**, the named maintainer/operator and independen reader audience, a bounded evidence handoff, and no affected public claim establishe
Acceptance coverage	Mission	`artifacts/preflight/candidate-evidence-handoff.json`; this page; `journeys/candidate-evidence-handoff-manifest.json`	AC-1, AC-2, and AC-3 each have a concrete artifact and one non-exploratory journey whose `acceptance_checks` and `traces_to` sets agree
Smoke manifest and checker	Platform-owned reader-gate boundary	`tests/smoke_manifest.agent-orch.json`; `tests/check_documentation_smoke.py`; `artifacts/preflight/candidate-evidence-handoff.json`	One supplied reader attempt for `journey.resolve-reader-journey-harness-decision` used `python3 tests/check_documentation_smoke.py`, exited and observed `Operation not permitted` because localhost socket access was denied
Preserved product-contract validator evidence	Platform-owned validator authority	`preserved-validator-evidence.json` supplied by the Agent-Orch run	Three preserved records each report exit status `0`, passed `true`, `10` passed, `0` failed, and `0` skipped. Hashes, in supplied order, are `a0cc3d856871267cc247dab9c472f657558b1e253efe31cc2515a25badda9d2` `9347adc57824c6378b15a155a5f3a716e14588d7549d28a5cde2c485c9a69a` and `63699490ab521d98b12e507f2b11c252e0fd2f6476dd8747ece84cfcd6516f`
Page, registry, and journey outputs	Author, subject to the product contract	`content/public/candidate-evidence-handoff.md`; `registries/pages.json`; `journeys/candidate-evidence-handoff-manifest.json`	Metadata and registry identity agree; the pre-existing required headings are retained a the provenance matrix and decision headings are added; AC-1, AC-2, and AC-3 traces and the two non-empty allowlisted commands agree

For a handoff that is not yet complete, replace placeholders with explicit `unresolved` values and keep the original observation beside any later result. The minimum evidence record includes the artifact path, authority boundary, source reference, command, exit status, observed output, output location, acceptance trace, and blocker owner or `owner unresolved`. The preserved validator results are independent evidence: each supplied record's `10 passed`, `0 failed`, `0 skipped`, exit status `0`, command, and evidence hash must be copied only from that preserved validator artifact. The three records are retained separately because they support different upstream acceptance relationships; none is a new local validator claim. The smoke result is recorded from the permitted reader-gate environment, where the one localhost denial is unresolved. A product page may explain these distinctions, but it cannot certify platform-owned execution.

Candidate-Evidence Contract Provenance Matrix

This matrix maps each material packet claim to the source or evidence boundary that owns it, the deterministic route that can check it, the coverage actually visible in the supplied records, and a classification. A declared command is a route, not proof of execution; a platform-owned attestation is not replaced by an author assertion. “Not rerun” records the boundary of this handoff and does not turn an unobserved result into a pass.

Candidate-evidence claim or contract obligation	Authoritative source or evidence boundary	Existing deterministic test or validation route
Internal Knowledge scope, audience, bounded outcome, success hypothesis, and validation journey	`registries/workstream-selection-receipt.json` ; `journeys/candidate-evidence-handoff-manifest.json`	Manifest journey `journey.candidate-evidence-handoff-preflight` ; decla `no:cacheprovider tests/test_product_contract.py`
Sealed authority revision `2026-08-08` , exactly three authority-listed source IDs, and `employee_may_modify: false`	`sources/authority.json` ; preserved Agent-Orch validator evidence	Preserved `PYTHONDONTWRITEBYTECODE=1 python3 -m json.tool sources/`
The three authority-listed IDs have public-safe source-registry records with their recorded class, locator, revision, and digest	`registries/sources.json` joined to `sources/authority.json`	Existing packet's explicit source-boundary preflight command; `python3 -m json.` validates the preserved authority record
Internal Knowledge is the selected queue and the exact three-queue comparison remains visible before item impact or feasibility	`registries/workstream-selection-receipt.json`	Candidate preflight journey declaration; declared `python3 -B -m pytest -p no tests/test_product_contract.py` route
Candidate packet acceptance coverage is exactly AC-1 through AC-3, with one required non-exploratory journey and matching `acceptance_checks` and `traces_to` for each	`journeys/candidate-evidence-handoff-manifest.json`	`tests/test_candidate_evidence_preflight.py::test_manifest_match` `::test_manifest_model_and_fixture_share_journey_declarations`
The candidate handoff allowlist and each journey command are exact, non-empty strings	Candidate handoff manifest `command_allowlist` and journey declarations	`tests/test_candidate_evidence_preflight.py::test_allowlist_is_e` `::test_command_claims_are_not_normalized_before_allowlist_match` `::test_allowlisted_command_must_match_the_declared_journey_comm`
Evidence records retain exact names, traces, steps, source references, command claims, exit codes, observed results, and the independent review-verdict requirement	`docs/candidate-evidence-preflight-contract.md` ; candidate manifest; platform-owned review boundary `code-reviews/content-review.verdict.json`	`tests/test_candidate_evidence_preflight.py` fixture, schema, CLI, and

Candidate-evidence claim or contract obligation	Authoritative source or evidence boundary	Existing deterministic test or validation route
The documentation smoke route and its supplied reader result remain distinct	`tests/smoke_manifest.agent-orch.json`; `tests/check_documentation_smoke.py`; platform-owned reader-gate evidence	Candidate manifest route `python3 -B tests/check_documentation_smoke.p`
Existing packet text retains three platform-owned product-contract validator records, each reported as exit `0`, `10` passed, `0` failed, and `0` skipped, with distinct hashes	`preserved-validator-evidence.json` supplied by the Agent-Orch handoff; validator authority remains platform-owned	Existing packet's copied `PYTHONDONTWRITEBYTECODE=1 python3 -m pytest tests/test_product_contract.py` claims
Page metadata, page registry identity, journey references, and candidate-only transition boundaries remain coherent	This page; `registries/pages.json`; candidate manifest; receipt `forbidden_actions`	`python3 tools/validate_content.py`; `tests/test_product_contract` release tests

Mismatch Classification and Lee Decision

The matrix separates four states. Confirmed alignment means the source declarations agree, not that an unrun command passed. Evidence gaps identify a missing artifact or an authority boundary not covered by the preserved result. Source/test mismatches identify declarations that do not line up with the test route currently named. Genuinely unresolved choices are limited to the one operator decision below; they do not authorize a transition.

Confirmed alignment. The preserved authority command is successful, the authority JSON retains revision `2026-08-08`, the three listed source IDs, and `employee_may_modify: false`, and the workstream receipt and candidate manifest keep the Internal Knowledge, candidate-only scope visible. The packet's provenance claims therefore remain bounded to the sources and evidence records named above.

Evidence gaps. The preserved result does not test the authority-to-source registry join, the page-to-registry join, the candidate page's review-verdict artifact, or the prior product-contract records. The named review path is a requirement, not proof that `code-reviews/content-review.verdict.json` exists or is clean. Those gaps keep handoff completion unresolved.

Source/test mismatches. The candidate handoff manifest declares three candidate journeys and two `python3 -B` commands, while the current `journeys/user_journeys_manifest.json` is a different publication-gap manifest with different journey names and four JSON commands; the candidate test file explicitly compares those manifests. The general preflight contract also describes its own synchronized five-check route. Separately, the supplied smoke attempt omitted `-B` relative to the candidate allowlist. These are declaration and evidence-routing mismatches to reconcile at the owning source boundary, not evidence of a product harness defect.

Genuinely unresolved choice for Lee. Retain this as a candidate-only, environment-specific evaluation blocker and route one permitted follow-up in an environment with localhost socket access. Do not name a recurring product harness defect, affected claim, or repair target from the single `Operation not`

`permitted` attempt. No further decision is needed here about approval, promotion, publication, scheduling, rollback, a release pointer, or a live-directory write; those controls remain outside this packet.

Acceptance Checks

- **AC-1 — Acceptance coverage and authority:** The reader verifies the **Internal Knowledge** workstream, audience, intended outcome, success hypothesis, validation journey, sealed authority revision `2026-08-08`, exactly three authority-listed public-safe source IDs, and `employee_may_modify: false`. The reader also confirms that AC-1, AC-2, and AC-3 each have a non-exploratory natural-language journey trace before checklist drafting, without copying private governance material or inventing authority.
- **AC-2 — Evidence inventory and ownership:** The reader records every required artifact with its owning authority, source reference, observed validation result, exact command, output location, and unresolved blocker. The inventory includes the smoke manifest and checker, all three preserved validator records with their distinct hashes, and the page/registry/journey outputs. Product conclusions remain separate from platform-owned attestations; the localhost-denied `Operation not permitted` observation is an environment-specific unresolved blocker, not evidence of recurrence, an affected claim, or a repair target.
- **AC-3 — Re-executable documentation validation:** The documentation- validation journey contains the non-empty allowlisted command `python3 -B tests/check_documentation_smoke.py`. The reader can locate the supplied result for `journey.resolve-reader-journey-harness-decision`, its claimed command `python3 tests/check_documentation_smoke.py`, exit status `1`, and `Operation not permitted` output, then route a permitted follow-up in an environment with localhost socket access. The handoff stays candidate-only and performs no approval, activation, promotion, publication, scheduling, rollback, pointer change, or live-directory write.

Candidate Evidence Navigation Guide

Audience: workflow operators, independent reviewers, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Candidate Evidence Navigation Guide

Candidate-only Status

This guide and its companion records are candidate-only evidence for workflow operators and reviewers. They are inspectable product artifacts, not published authority, not approval, and not a release pointer. Candidate-only status does not establish promotion, publication, scheduling, rollback, activation, or a write to a live directory. Keep the package parked until the responsible review and platform-owned decisions are supplied separately.

Find the Evidence

Start with `sources/authority.json`. Read its revision, its three logical source IDs, and its `employee_may_modify` value, then match those IDs to the public-safe records in `registries/sources.json`. Next compare this page's metadata with its entry in `registries/pages.json`, and open `journeys/user_journeys_manifest.json` to follow the three acceptance traces. The candidate record at `releases/candidate-evidence-navigation.json` ties those product artifacts together. The preserved Agent-Orch validator record is platform-owned captured evidence: its exact command, exit status, pass status, and hash must be read from the supplied record rather than replaced by an author assertion or a validator rerun.

Use the source registry for provenance, the page registry for identity, the journey manifest for reader procedure, and the release record for status and boundary facts. A declared command is an allowlist entry, not proof that the command ran. If a fact is absent from its owning record, mark it unresolved; do not fill the gap with plausible prose or private governance material.

Review Handoff

The operator hands off the Markdown page, synchronized page entry, journey manifest, source references, and candidate release record as one inspectable packet. The author records product-side evidence and unresolved gaps. An independent reviewer checks the headings, metadata, source references, journey traces, and candidate-only status on a separate review path. Neither the author's summary nor a product conclusion self-certifies platform execution, human approval, promotion, publication, scheduling, rollback, or a release pointer. Preserve the exact supplied validator evidence alongside any later reader or review record, including failures or unavailable observations.

The handoff is complete for navigation when a reviewer can locate each source, map each AC-1 through AC-3 check to a natural-language journey, and distinguish declared commands from observed results. Completion of this guide does not request or perform a transition; it only makes the candidate evidence easier to inspect and route to the owner of the next decision.

Source and Status Boundaries

The sealed source boundary is the authority for the guide's semantic provenance. `sources/authority.json` lists `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`; the matching registry records are the available public-safe source metadata. The guide uses no registered claim reference, so it does not add a semantic product claim. The governing repository remains the owner of controls that are outside these product records.

The candidate record describes what was authored and what remains open. The declared inputs establish no prior-version identity or rollback target for this new guide; those facts are recorded as null and unresolved rather than guessed. No publication pointer is created or changed. Candidate evidence can support a bounded product review, but it cannot become published authority, a live directory result, or proof that a platform-owned gate or transition occurred.

Acceptance Checks

- **AC-1 — Source and identity trace:** The reader opens `sources/authority.json`, verifies its recorded revision, three source IDs, and `employee_may_modify` value, matches the IDs to `registries/sources.json`, and confirms that this page's header and registry entry have the same id, title, visibility, audiences, reading modes, empty claim references, and source references.
- **AC-2 — Evidence navigation and review handoff:** The reader locates the page, registry, journey manifest, candidate release record, and supplied validator evidence; follows one non-exploratory journey for each acceptance check; and distinguishes a declared allowlisted command from the preserved validator's exact command, exit status, pass status, and evidence hash without claiming a rerun.
- **AC-3 — Candidate-only and release facts:** The reader confirms that the release record is candidate-only, unpublished, unpromoted, and has no release-pointer or live-directory action; reports prior-version and rollback-target facts as null and unresolved when no authoritative record establishes them; and does not infer approval, publication authority, or a transition from this packet.

Candidate Evidence Provenance

Audience: Lee, current company operators, future Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

id: page.candidate-evidence-provenance title: Candidate Evidence Provenance visibility: public candidate_visibility: candidate-only candidate_only: true audiences: - Lee - current company operators - future Internal Knowledge playbook authors - independent readers reading_modes: - executive - engineer - ai claim_refs: [] source_refs: - source.naming-decision - source.documentation-boundary - source.platform-delivery-contract

Candidate Evidence Provenance

Purpose

This is a bounded provenance packet for Lee and auto-orch operators working in the selected **Internal Knowledge** workstream. It explains what the candidate evidence actually establishes, where each statement comes from, which product tests are relevant, and which facts remain owned by a platform evaluator or require Lee. The packet contains evidence statements rather than a new semantic product claim, and its YAML header is synchronized with the page registry and the compatibility metadata consumed by the product validator.

The packet keeps four boundaries visible: sealed authority is not employee modifiable, a preserved validator result is not a new local run, an environment-denied reader attempt is not proof of a recurring product defect, and candidate-only product evidence is not approval or publication. Missing recurrence, product impact, or transition authority is recorded as a gap; it is never filled with guessed prose or an invented repair target.

Evidence Matrix

The rows below are candidate-evidence claims made by this packet, not entries in the governed semantic claims registry. Each row names the direct evidence owner and fields, then names the sealed-authority boundary and the exact page identity fields that make the statement traceable. The identity anchor is the same in every row: `content/candidate-evidence-provenance.md` has page ID `page.candidate-evidence-provenance`, title `Candidate Evidence Provenance`, public/candidate-only visibility, empty `claim_refs`, and the three source references, and the matching object exists in `registries/pages.json`. A product test can validate a contract or structure; it cannot replace platform-owned attestations or create missing runtime evidence.

| Candidate-evidence claim | Direct evidence owner and fields | Sealed-authority and page-identity trace | Relevant product coverage, mismatch, and Lee decision | | --- | --- | --- | --- | | **CE-1 — Authority boundary:** revision `2026-08-08`, exactly the three logical source IDs, and `employee_may_modify: false`. | Direct owner: `sources/authority.json`, fields `authority_revision`, `source_files`, and `employee_may_modify`; corroboration: `registries/sources.json`, each matching `id` and `disclosure`. | Sealed boundary anchor: `sources/authority.json` is employee-immutable; page identity anchor: the page header and `registries/pages.json` entry both carry the page ID, title, visibility, candidate-only fields, empty `claim_refs`, and the same three `source_refs`. | The product contract covers schema/reference and source-boundary structure, but neither test substitutes for the sealed authority. No mismatch is recorded; all three source records are public-safe. No product decision is requested. | | **CE-2 — Preserved validator:** six independent Agent-Orch validator records supplied for this handoff retain their exact commands, statuses, counts where supplied, summaries, durations, outputs, and hashes. | Direct owner: the preserved validator artifact supplied for this step, represented in `releases/candidate-evidence-provenance-candidate.json` under `preserved_validator_evidence.records`; fields are keyed by `source_step_id`, command, exit status, counts, duration, summary, output, and evidence hash. | The sealed authority constrains the product source boundary but does not attest validator execution. The page header and registry identity remain the same synchronized identity anchor for this evidence packet. | Records 1–3 and 5 are the exact `pytest` command with exit `0`, `10 passed`, and hashes `087109894e15960d623ee618239670a42955cf367c99d22a0c2f2c78223489e5`, `01cc6a08169e3a97d213417792d229c42152585980064a1d5bd6db9ba84a1e7e`, `593ce049b9a988035ad4c23589ff3cd4e08f85567eda7c2d00adf542cbece936`, and `50f61ea74d7894e238a52f31c0b1f05a8b97a3b67cdafb033a230d4994372a57`; records 4 and 6 are `tools/build_projection.py`, exit `0`, hashes `fe30b42ea3bd9d571bb0d29359163c3709c291c270cf426173a267060b438ee6` and

`dac432306c6aa04bdd06d2d3b9eb9520903f180531ef31070df8d43e2322acb0`. They are preserved evidence, not a local rerun, and Lee is not asked to treat them as approval or platform attestation. | | **CE-3 — Reader-gate observation:** one attempt of `journey.resolve-reader-journey-harness-decision` exited `1` with `Operation not permitted` because localhost socket access was denied. | Direct owner: the platform-owned evaluator record described by `questions-for-the-human.md#blocking-question` and the read-only `journeys/reader_journeys_manifest.json`; fields are journey ID, exact command, attempt count, exit status, output, and environment context. | The sealed authority remains the source-boundary anchor, while the page header and registry entry identify the exact candidate page to which this observation is attached. Neither identity anchor turns the observation into a product claim. | Evidence gap, not a confirmed product mismatch: the single environment-specific failure establishes no recurrence, retry count, affected claim, or repair target. The product journey-fixture test covers structure only. Lee must retain the gap and route the exact command for permitted follow-up. | | **CE-4 — Candidate-only boundary:** the record is unpublished and unpromoted, with no pointer, scheduling, rollback, activation, or live-directory write. | Direct owner: `releases/candidate-evidence-provenance-candidate.json` fields `status`, `candidate_only`, `published`, `promoted`, and `publication_boundary`, with `AGENTS.md` as the local boundary contract. | The sealed authority is unchanged; the synchronized page identity is the header/registry pair named above. The candidate record's `content_identity` repeats that pair and names the journey manifest, so the boundary facts cannot be detached from the page. | No mismatch is claimed. The absence of a prior approved version and rollback target is explicitly `not_established`, not a failed lookup to repair. No transition decision is requested; approval, promotion, publication, scheduling, rollback, and activation remain separately governed. | | **CE-5 — Workstream context:** this bounded packet belongs to the selected **Internal Knowledge** workstream and serves Lee, current operators, playbook authors, and independent readers. | Direct owner: `questions-for-the-human.md#affected-work` and `registries/workstream-selection-receipt.json`; synchronized context is repeated in the page header, registry entry, candidate `content_identity`, and journey manifest. | The sealed authority is the immutable source-boundary context, not a workstream selector. The synchronized page identity proves which packet carries the context; `claim_refs: []` keeps it from becoming a new governed semantic claim. | No contradiction is supplied; this is context, not a fresh scheduling result. The product suite does not reselect the workstream. No decision is requested from Lee about scheduling or transition. |

Mismatch Status

The product-side source and identity facts are aligned: the page, registry entry, candidate record, and journey manifest use the same page identity, title, visibility, audience, reading modes, claim references, and logical source references. All six preserved validator records are retained with the exact fields supplied by the authoritative artifact; the four pytest records include their counts and summaries, while the two projection records have no test-count claim. These are evidence alignments, not a claim that this step executed any recorded validator or that a platform gate passed.

One evidence gap remains intentionally open. The platform-owned reader-gate record contains one denied attempt with exit status `1` and output `Operation not permitted`; it does not contain recurrence, a retry count, an affected claim, or a named repair target. Therefore the mismatch status is **environment-specific evaluation blocker; product defect not established**. That status must not be rewritten as a product harness failure, and the reader manifest is a traceability input rather than a repair target. The absence of a prior approved version or rollback target is separately recorded as an honest absence, not as an evidence mismatch.

Unresolved Decisions

The only decision genuinely requiring Lee in this packet is bounded and reversible: retain the candidate-only evidence blocker as unresolved and route a permitted follow-up evaluation in an environment with the required localhost reader service and socket access. A later authoritative reader record must provide the journey ID, exact allowlisted command, exit status, observed output, retry count or recurrence evidence, source references, and affected work item before anyone names a product defect or repair target. This is a decision about where evidence should be routed, not permission to change the manifest, authority, claims, governance controls, or platform gates.

Lee is not being asked to approve, promote, publish, schedule, activate, roll back, create a release pointer, or write a live directory. No decision is inferred from the preserved validator result, the workstream label, the page's public visibility, or the existence of this candidate file. Until authoritative follow-up evidence and any separately governed decision arrive, the smallest safe disposition is to leave the candidate parked and unchanged.

Acceptance Checks

- **AC-1 — Source and identity trace:** The reader verifies the authority revision `2026-08-08`, the exact three authority-listed logical source IDs, their public-safe registry records, `employee_may_modify: false`, and the synchronized page header and registry identity without copying private governance material or asserting a new semantic claim.

- **AC-2 — Preserved validator trace:** The reader reports all six preserved records supplied for this handoff: the four pytest records from `step_01_seal_authority`, `step_03_author_provenance_packet`, and the two `step_04_repair_and_verify_packet` attempts, plus the two `step_05_generate_projection` records. The pytest records retain their exact command, declared bytecode-disabled command, exit `0`, pass status, `10` passed with `0` failed and `0` skipped, durations `3.150056`, `3.205681`, `3.18602`, and `3.298212` seconds, summaries `10 passed in 2.60s`, `10 passed in 2.64s`, `10 passed in 2.62s`, and `10 passed in 2.74s`, empty stderr, and hashes `087109894e15960d623ee618239670a42955cf367c99d22a0c2f2c78223489e5`, `01cc6a08169e3a97d213417792d229c42152585980064a1d5bd6db9ba84a1e7e`, `593ce049b9a988035ad4c23589ff3cd4e08f85567eda7c2d00adf542cbece936`, and `50f61ea74d7894e238a52f31c0b1f05a8b97a3b67cdafb033a230d4994372a57`; the projection records retain `tools/build_projection.py`, passed status with exit `0`, empty output, and hashes `fe30b42ea3bd9d571bb0d29359163c3709c291c270cf426173a267060b438ee6` and `dac432306c6aa04bdd06d2d3b9eb9520903f180531ef31070df8d43e2322acb0`. No record is rerun or replaced.
- **AC-3 — Evidence-gap classification:** The reader records the one journey ID, exact command, exit status `1`, and `Operation not permitted` output with its denied-localhost context, and distinguishes that environment-specific evaluation blocker from an unestablished recurrence, affected claim, product defect, or repair target.
- **AC-4 — Lee decision and candidate boundary:** The reader identifies Internal Knowledge and the packet audience, states Lee's smallest decision to retain the unresolved blocker and route permitted follow-up evidence, confirms the honest absence of prior-version and rollback facts, and confirms that no approval, promotion, publication, scheduling, activation, rollback, pointer, or live-directory action occurred.

This packet is complete only as a candidate-evidence provenance record. A future evaluator or decision owner must supply any missing runtime evidence or transition authority; this page does not manufacture either one.

Candidate Evidence Provenance: Bounded Repair Route

Audience: Lee, current company operators, agent-orch maintainers, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Candidate Evidence Provenance: Bounded Repair Route

Scope and sealed authority

This candidate-only packet records the deterministic candidate-evidence preflight failure and the smallest supported repair route. It is product-side provenance, not a new semantic claim, platform attestation, approval, or publication decision.

The sealed authority is `sources/authority.json`. It has revision `2026-08-08`, lists exactly these logical sources, and sets `employee_may_modify` to `false`:

- `source.naming-decision`
- `source.documentation-boundary`
- `source.platform-delivery-contract`

The matching records in `registries/sources.json` are public-safe. This packet does not copy private governance material and does not alter the authority or its source boundary.

Preserved validator evidence

The independent Agent-Orch validator preserved the following result for this step. It is not a local rerun:

- Command: `python3 -m pytest -p no:cacheprovider tests/test_candidate_evidence_preflight.py -q`
- Exit status: `1`
- Duration: `0.896599` seconds
- Result: `16 failed, 6 passed, 8 subtests passed`
- Evidence hash: `2980064346d86791b6305f3eceb949f8b6edd727d6ec1d9fac8e34c6ace9bd3b`

The preserved output shows declaration and shape mismatches, including a manifest/model allowlist mismatch, missing candidate-handoff journey IDs in the manifest, and the absent `evidence_schema` expected by the fixture oracle. The preserved record is the authority for those observed results. This step does not replace it with a new command result.

What the failure establishes

The test fixture is shaped for the pinned candidate-evidence handoff: its journeys are the three `journey.candidate-evidence-handoff-*` declarations and its claimed commands are the two commands in `models/candidate_evidence_handoff_contract.json`. The current `journeys/user_journeys_manifest.json` instead declares a publication-reader gap manifest with four JSON-tool commands and no candidate-evidence schema. That is why the independent oracle cannot find the fixture's first journey and why the model and manifest do not synchronize.

This establishes a bounded declaration-reconciliation defect in the candidate-evidence route. It does not establish a change to sealed authority, a recurring reader-harness defect, an affected product claim, or a runtime repair target. The separate `Operation not permitted` reader observation remains an environment-specific evaluation blocker unless authoritative follow-up evidence establishes more.

Bounded repair route

The owning maintainer should reconcile the candidate-evidence manifest and model at their source boundary before treating a candidate-evidence handoff as complete:

1. Compare `journeys/user_journeys_manifest.json` with the pinned `journeys/candidate-evidence-handoff-manifest.json` and preserve the pinned three journey IDs, exact names, acceptance mappings, expected exit codes, review-verdict path, evidence schema, and canonical allowlist.
2. Synchronize the corresponding execution declarations in `models/candidate_evidence_handoff_contract.json` and retain the fixture's exact journey names, traces, command claims, exit codes, and observed results. Do not normalize near-match command strings.
3. Keep the required independent review artifact separate from reader evidence; naming `code-reviews/content-review.verdict.json` is not proof that the artifact exists or passes.
4. After an authorized source-boundary repair, run the exact preflight in an environment permitted to create no bytecode or pytest cache files, and preserve its complete result. A future pass must not be inferred from this preserved failure.

This packet and its registry/journey projection are the only product-side corrections made in this bounded step. The authority, source registry, pinned handoff manifest, model, fixtures, tests, governance controls, and publication boundary remain unchanged here. No unresolved canonicalization or policy question is decided for Lee.

Candidate-only boundary and Lee decision

The selected workstream context is **Internal Knowledge**. The smallest safe disposition is to retain the candidate-only handoff as unresolved and route the declaration repair to its owning manifest/model boundary. Lee is not asked to approve, promote, publish, schedule, activate, roll back, change a release pointer, or write a live directory.

The later authoritative record must show synchronized declarations and a readable independent review verdict before handoff completion is considered. If a reader-gate follow-up is also needed, it must retain the exact journey ID, command, exit status, observed output, retry or recurrence evidence, source references, and affected work item. Missing recurrence or product impact must remain unresolved.

Acceptance checks

- **AC-1 — Authority trace:** The reader verifies revision `2026-08-08`, the exact three authority-listed source IDs, their public-safe registry records, and `employee_may_modify: false` without modifying authority.
- **AC-2 — Preserved failure trace:** The reader reports the preserved validator command, exit status `1`, duration, result counts, and evidence hash without claiming a rerun or replacing the captured record.
- **AC-3 — Bounded repair route:** The reader can identify the manifest/model synchronization defect, the exact declaration fields to reconcile, and the separate review-verdict requirement without inventing a product defect or policy decision.
- **AC-4 — Candidate boundary:** The reader keeps the Internal Knowledge handoff unresolved and candidate-only, with all approval and publication transitions untouched.

Candidate-Evidence Discrepancy Triage

Audience: agent-orch maintainers, release reviewers | Mode: executive, engineer, ai | Provenance: Candidate packet candidate-evidence-discrepancy-triage (candidate-evidence-discrepancy-triage.json, step: step_03_author_triage_report)

---

id: page.candidate-evidence-discrepancy-triage title: Candidate-Evidence Discrepancy Triage visibility: public candidate_visibility: candidate-only candidate_only: true audiences: - agent-orch maintainers - release reviewers reading_modes: - executive - engineer - ai claim_refs: [] source_refs: - source.naming-decision - source.documentation-boundary - source.platform-delivery-contract

---

Candidate-Evidence Discrepancy Triage

Purpose

This is a read-only candidate-evidence triage for agent-orch maintainers and release reviewers. It maps declaration drift, fixture rejection cases, and the supplied platform observations to the exact manifest, model, fixture, authority owner, reproducible check, and escalation owner. It distinguishes a shape rejection from a runtime observation and a product conclusion from a platform attestation. The page is candidate-only: it creates no approval, release pointer, promotion, publication, schedule, rollback, activation, or live-directory effect. Missing recurrence, product impact, or authority is kept unresolved rather than completed with plausible prose.

Evidence Matrix

The matrix is a traceability aid, not a resolution ledger. The manifest named in every row is `journeys/user_journeys_manifest.json`; the pinned comparison manifest is `journeys/candidate-evidence-handoff-manifest.json`; and the machine-readable model is `models/candidate_evidence_handoff_contract.json`. The human-readable contract at `docs/candidate-evidence-preflight-contract.md` remains a separate five-check declaration. The three-check manifest delivered with this packet is synchronized to the executable/model route, but that synchronization does not canonize or resolve the five-versus-three declaration discrepancy.

Discrepancy and observed fact	Manifest	Model
D-1 — Declaration cardinality and route drift: the human contract names five checks and five canonical journeys, while the executable handoff route names AC-1 through AC-3 and three journeys.	`journeys/user_journeys_manifest.json`; compare `journeys/candidate-evidence-handoff-manifest.json`.	`models/candidate_evidence_handoff_contract.jso` including its three pinned journey declarations.
D-2 — Journey-name mismatch: `invalid-journey-name.json` substitutes a shortened name for a pinned journey name.	The exact `name` and `traces_to` values in `journeys/user_journeys_manifest.json`.	The model's `journeys` array pins the byte-for-byte name.
D-3 — Command allowlist mismatch: `disallowed-command.json` claims a command outside the exact allowlist.	The manifest allowlist contains only `python3 -B tests/check_documentation_smoke.py` and `python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py`.	The model repeats that exact allowlist and the journey comman bindings.
D-4 — Missing observed result: `malformed-evidence.json` omits the required command observation, so a declaration cannot be mistaken for execution evidence.	The manifest's evidence schema requires `observed_result` for each command claim.	The model requires `command`, `exit_code`, and `observed_result`.

Discrepancy and observed fact	Manifest	Model
D-5 — Independent-review path gap: `missing-review-path.json` omits the required review-verdict path, while the manifest requires `code-reviews/content-review.verdict.json`.	`journeys/user_journeys_manifest.json` requires the path and accepted verdict shape.	The model repeats the same required review path and verdict constraints.
D-6 — Reader-gate command/result mismatch: the supplied platform observation records `python3 tests/check_documentation_smoke.py`, exit `1`, and `Operation not permitted`, while the pinned journey declares `python3 -B tests/check_documentation_smoke.py`; localhost access was denied.	The third journey in `journeys/user_journeys_manifest.json` declares the exact `-B` command, expected exit `1`, and AC-3 trace.	The model's third journey carries the same command and expected exit.

Discrepancy and observed fact	Manifest	Model
D-7 — Preserved-validator representation boundary: the authoritative records supply `python3 -m pytest -p no:cacheprovider tests/test_product_contract.py` as their command array and separately record the bytecode-disabled declared command, while the journey declaration uses the exact `-B` form.	The first two journeys in `journeys/user_journeys_manifest.json` declare the `-B` form and expected exit `0`.	The model carries those exact journey declarations.

Discrepancy Triage

Triage starts by classifying the row, not by assigning a repair. D-1 is a declaration conflict between the five-check human contract and the pinned three-check executable/model route. The delivery keeps the user manifest on the pinned three-check route because the executable preflight explicitly pins that shape, while leaving the human-contract conflict open for its owner. D-2 through D-5 are intentionally negative fixtures: they show how the preflight rejects altered identity, unallowlisted commands, missing observations, and a missing review path. Their rejection shape must not be reported as a live reader failure or a resolved product defect.

D-6 is the only supplied runtime-style discrepancy. One platform-owned reader-gate attempt exited `1` with `Operation not permitted` because the evaluator environment denied localhost socket access. The command in that record lacks the manifest's `-B` spelling, so the record must be preserved as observed rather than silently rewritten. One denied attempt does not establish recurrence, a retry count, an affected claim, a product harness defect, or a repair target. D-7 records the same kind of boundary for the five preserved validator results: each is authoritative captured evidence, but each command array and declared command must not be conflated with the manifest's reader claim. No discrepancy in this report is called resolved without current owner-supplied evidence that proves the relevant fact.

Ownership and Escalation

The author owns only this product-side packet: the candidate page, its registry identity, the candidate change record, and the reader-journey manifest. The author may describe evidence and unresolved gaps, but may not change `sources/authority.json`, private governance, the validator authority, the governing contract, or a platform-owned reader record. The agent-orch contract owner owns reconciliation of the five-check versus three-check declarations, exact journey identity, and allowlist changes. A change that would alter governing authority, routing, validator authority, activation, or publication policy requires the separate governance authority; it is not an authoring decision.

The platform evaluator owns the reader-gate execution, environment cause, and any permitted follow-up result. The system-validator authority owns the preserved product-contract record and its evidence hash. The independent release reviewer owns the verdict at the required review path; the presence of a path in a manifest is not a verdict. Lee's bounded human decision is whether to retain the unresolved environment-specific blocker and route a permitted follow-up evaluation. Lee is not being asked by this page to approve, promote, publish, schedule, activate, roll back, create or change a release pointer, or write a live directory. Until the named owner supplies the missing evidence, the candidate stays parked and the discrepancy stays open.

Acceptance Checks

- **AC-1 — Traceability:** A maintainer or release reviewer can start at this page and map every listed discrepancy to `journeys/user_journeys_manifest.json`, the pinned handoff manifest, `models/candidate_evidence_handoff_contract.json`, the named fixture or preserved artifact, the three public-safe source IDs, and an explicit owner without treating a declaration as a run result.
- **AC-2 — Reproducibility:** A reviewer can replay the exact non-empty commands declared by the user manifest, compare names and commands byte-for-byte, and preserve the supplied exit status, output, counts, and evidence hashes. All five preserved validator results are read as authoritative evidence and are not rerun or replaced in this handoff.
- **AC-3 — Authority ownership:** Each discrepancy has a named authority and escalation route; Lee and governance authority decisions are distinguished from author work, independent review, platform evaluation, and validator attestations. The reader retains the candidate-only boundary and leaves recurrence, affected claims, repair targets, approval, promotion, publication, scheduling, rollback, pointers, and live-directory actions unresolved unless separately proved.

This report is complete as a candidate-only triage record, not as a repair, review verdict, release decision, or publication authorization. A later owner may add evidence to the relevant route, but may not rewrite the original observation to make the matrix appear closed.

Candidate-Only Reader Evidence Preflight Guide

Audience: auto-orch operators preparing governed-run evidence, Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Candidate-Only Reader Evidence Preflight Guide

Purpose and Scope

The primary workstream is **Internal Knowledge**. This guide is for auto-orch operators preparing governed-run evidence, Internal Knowledge playbook authors, and independent readers. Its outcome is a candidate-only reader-evidence packet that a later evaluator can inspect and reproduce: the operator can name the reader problem, intended outcome, exact journey, source or authority, artifact location, acceptance trace, and unresolved gaps without turning documentation into a platform attestation. The guide covers the preflight before or during a governed run; it does not alter sealed authority, the source registry, or an evaluator's independent evidence.

Use the guide to prepare evidence for a bounded reader check, not to make a transition decision. A command declaration, page entry, expected result, or product conclusion is useful input, but none is proof that a command ran or that a gate passed. Keep missing facts explicitly unresolved and keep product evidence separate from platform-owned validator, identity, route, approval, activation, or publication evidence.

Reader-Evidence Preflight

Before checklist authoring, write down the selected **Internal Knowledge** workstream, the operator audience, the reader problem, and the intended outcome. Open the synchronized page header, page registry entry, source authority, source registry, and journey manifest. Confirm that the page has no claim references and uses only the three approved public-safe source refs. Then enumerate AC-1 through AC-4 and match each criterion to one required non-exploratory journey. A criterion without a journey, authority, or artifact location is an unresolved coverage gap, not an invitation to infer completion.

For each journey, prepare one evidence row and reserve space for the exact journey name, the eligible non-empty command prefix, observed exit code, reproducible output, source or authority, artifact location, acceptance trace, and failed-evidence record. The allowlisted command in this guide is `python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py -q`. It is a declaration to be used by an authorized reader; it is not an observed execution. Replace `unresolved` only with evidence captured by that reader.

Command-Claim Checklist

Complete exactly one row for every required journey in `journeys/candidate-only-reader-evidence-preflight.json`. The status must be `completed` only when the reader record contains the requested observation; otherwise use `unresolved` and name what is missing. The exact allowlisted invocation is both the eligible non-empty command prefix and the command claim for each row. Do not report a command as run merely because it appears here or in the manifest.

Status (`completed` or `unresolved`)	Exact journey name	Eligible non-empty command prefix / exact command claim	Observed exit code and reproducible output	Source or authority	Artifact location	Ac tra
unresolved	As an auto-orch operator, confirm the Internal Knowledge scope and intended reader outcome before a governed run.	`python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py -q`	unresolved — capture the exit code and output from the governed reader record; a declaration is not execution proof.	`sources/authority.json` , `registries/sources.json` ; mission/author journey authority	unresolved — record the governed-run evidence path	AC
unresolved	As an Internal Knowledge playbook author, inventory each reader-evidence field and its owning source or authority.	`python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py -q`	unresolved — capture reproducible output and distinguish missing evidence from a passing result.	This guide, `registries/pages.json` ; author journey authority	unresolved — record the completed inventory path	AC
unresolved	As an independent reader, verify the exact command claim and acceptance trace without inferring execution.	`python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py -q`	unresolved — capture the observed exit code and reproducible output; do not substitute an expected result.	`tests/test_product_contract.py` , this guide; mission journey authority	unresolved — record the independent reader record path	AC
unresolved	As an auto-orch operator, confirm that the evidence packet remains candidate-only and	`python3 -B -m pytest -p no:cacheprovider tests/test_product_contract.py -q`	unresolved — capture the bounded reader result without treating it as approval or a transition attestation.	`sources/authority.json` , this guide, `registries/pages.json` ; mission journey authority	unresolved — record the candidate-only evidence path	AC

Status (`completed` or `unresolved`)	Exact journey name	Eligible non-empty command prefix / exact command claim	Observed exit code and reproducible output	Source or authority	Artifact location	Ac tra
	preserves every unresolved boundary fact.					

The checklist records evidence claims, not authority to act. It must not infer approval, promotion, publication, scheduling, rollback, a release pointer, activation, or a live-directory action from a command, an exit code, a reader verdict, or a candidate record. If the output is absent, keep the row unresolved; if execution failed, keep the failure and its artifact location available for independent review.

Candidate-Only Boundary

Candidate-only means that the guide and its reader evidence describe a reversible product-side handoff awaiting separately governed decisions. It is not approval, promotion, publication, scheduling, rollback, a release pointer, activation, or permission to write a live directory. The manifest's journeys can establish what a reader should inspect and what product evidence was observed, but they cannot create platform-owned attestations or change the governing repository's authority contract.

Keep the candidate disposition explicit in every operator summary. Separate the product conclusion from platform evidence about execution, evaluator route, identity, gate state, or human authorization. If previous-version identity, rollback-target facts, or a transition record is not supplied by its owner, write `unresolved`. A synchronized page registry proves metadata alignment only; it does not establish a release pointer or a live-directory change.

Failed-Evidence Preservation

A failed command, denied socket, missing artifact, nonzero exit code, or environment-specific observation is evidence about what was observed. Preserve the exact journey name, exact command, exit code, reproducible output, source or authority, artifact location, time or run identifier when available, and acceptance trace. Keep the failed record beside any later retry or successful result so an independent reader can distinguish attempts. Do not delete, rewrite, or silently replace failed evidence with a declaration that the command was eligible.

The operator may mark a row `unresolved` when the result or owner is missing. That label is honest evidence status, not a product defect and not a reason to invent a repair target. A preserved failure also does not prove recurring harness failure, approval, promotion, publication, scheduling, rollback, activation, or a live-directory action. Route a permitted follow-up evaluation to the authority that owns the missing fact while retaining this candidate-only boundary.

Acceptance Checks

- **AC-1 — Scope and outcome:** The reader identifies Internal Knowledge as the primary workstream, names the operator audience, states the reader problem and intended outcome, and traces that coverage to the required operator journey without claiming that a command ran.
- **AC-2 — Evidence inventory:** The reader can complete or mark unresolved every checklist field: exact journey name, eligible non-empty command prefix, observed exit code and reproducible output, source or authority, artifact location, acceptance trace, and preservation of failed evidence.
- **AC-3 — Command claim boundary:** The reader verifies that every journey uses the exact allowlisted command and distinguishes that declaration from observed execution. Missing output, a nonzero exit code, or a failed attempt remains visible and does not become an approval or product conclusion.
- **AC-4 — Candidate-only handoff:** The reader confirms that the packet is candidate-only and does not infer approval, promotion, publication, scheduling, rollback, a release pointer, activation, or a live-directory action. Product conclusions and platform-owned attestations remain separate.

Acceptance is complete only when each required journey has one completed or unresolved row and every unresolved or failed fact is named for follow-up. The guide prepares evidence for a governed reader; it does not perform the reader run or authorize any later transition.

Contract-Derived Candidate Evidence Preflight Receipt

Audience: auto-orch maintainers, stage workers, Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Contract-Derived Candidate Evidence Preflight Receipt

Purpose

This is a read-only feasibility receipt for the Internal Knowledge workstream. It is written for auto-orch maintainers and stage workers, with Internal Knowledge playbook authors and independent readers able to use the same contract. Before candidate authoring, it inventories the authoritative contracts, required journey coverage, artifact schemas, command boundary, expected outputs, deterministic validators, preventable failures, blockers, and ownership needed to make a bounded feasibility decision.

The receipt records only facts established by the declared inputs and preserved validator evidence. A command declaration is not execution evidence, a product-contract result is not a platform attestation, and a missing artifact is not a passing result. Unresolved or inconsistent prerequisites remain visible. This page does not authorize a product change or any later governed transition.

Canonical Inputs

The sealed authority input is `sources/authority.json`, schema version `1`, revision `2026-08-08`. Its `source_files` set is exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and it records `employee_may_modify: false`. `registries/sources.json` supplies the matching registered records; each is marked `public-safe`. Their authority classes, locators, revisions, and digests may be compared but may not be rewritten here. The digest-backed references to the separate governing repository do not expose private governance material or grant this product repository control over it.

The contract inputs are `content/candidate-evidence-handoff-preflight.md`, `docs/contract-derived-candidate-evidence-preflight-receipt.md`, `models/evidence-artifact-glossary-contract.md`, `tests/test_product_contract.py`, and `tools/validate_content.py`. Repository guidance comes from `AGENTS.md`, `docs/project-onboarding.md`, and `questions-for-the-human.md`. The smoke boundary is described by `tests/smoke_manifest.agent-orch.json` and `tests/check_documentation_smoke.py`. The authored output set is this page, its journey manifest, and the matching page-registry entry; named paths are not treated as available evidence unless an authoritative record supplies them.

The glossary contract requires authority and reader-evidence artifacts to stay distinct from product conclusions and platform-owned attestations. It also names `artifacts/glossary-authoring/authority-seal.json` and `artifacts/glossary-authoring/evidence-artifact-glossary-journeys.json`, but neither companion artifact is a trusted or declared input for this step. That absence is carried into the feasibility result rather than silently repaired.

Feasibility Receipt

The authority-to-source comparison is feasible as a read-only inspection: all three authority-listed logical IDs are present in the registered source set, and all three registered records are public-safe. This establishes an input inventory, not permission to modify `sources/authority.json`, `registries/sources.json`, external governance references, or the employee boundary. Any later stale, missing, conflicting, or non-public-safe field must be recorded as unresolved.

The synchronized manifest is schema version `1`, kind `reader-journey-manifest`, and authority `mission`. It enumerates exactly AC-1, AC-2, and AC-3. Its required journey is named `Verify the candidate evidence preflight receipt`, is non-exploratory, and traces to every one of those criteria. The journey trace describes what an independent reader may check; it does not claim that this authoring step ran a reader journey.

The manifest command allowlist contains only the non-empty prefix `python3 -m pytest`. Each declared journey uses the exact command `python3 -m pytest tests/test_product_contract.py -p no:cacheprovider` and expects exit code `0`. Its expected observable result is a passing deterministic product-contract validation, preserved separately from reader execution. The manifest includes no smoke command, shell wrapper, alternate pytest prefix, or claim that an authoring worker executed the reader route.

The preserved validator records supplied for this handoff report successful deterministic checks: semantic-only content validation, JSON syntax validation, the product-contract pytest command, and projection building each have exit status `0` and passed status in their preserved records. For the exact pytest record, validator identity `agent_orch.validators.command_succeeds@1` reports exit status `0`, passed `true`, and evidence hash `f7b97990b624e6b6124e11aa3452aff97720e8d82a216622e752986193ff58f2`. These are supplied platform-owned validator observations, not commands run or self-certified by this authoring step.

Expected authored outputs are the Markdown receipt at `content/contract-derived-candidate-evidence-preflight-receipt.md`, the synchronized reader contract at `journeys/contract-derived-candidate-evidence-preflight-receipt.json`, and one matching entry in `registries/pages.json`. The expected output is synchronization and explicit feasibility status; it is not a release, publication, or reader-execution artifact.

Blockers and Ownership

Blocker B-1 is a contract divergence. The candidate-evidence handoff page describes AC-1 through AC-4 and requires the `python3 -B -m pytest` prefix, while this mission's reader contract preserves AC-1 through AC-3 and the `python3 -m pytest` prefix. The acceptance sets and commands are not interchangeable. The authorized product-contract and reader-manifest maintainer owns reconciliation; the responsible individual is not identified by the declared inputs and remains unresolved.

Blocker B-2 is an artifact-schema gap. The glossary names the two `artifacts/glossary-authoring/` companion files, but their contents, digests, availability, and owner are not supplied in this step. The artifact-contract maintainer who can supply or explicitly withdraw those dependencies owns the gap; no individual identity is established. A path mentioned by prose cannot close this blocker.

Blocker B-3 is missing trusted reader-execution evidence. No trusted artifact input is declared for this handoff, and the authoring role cannot execute or self-certify the reader journey. The platform-owned evaluator must supply a record containing journey identity, exact command, exit status, observed output, source references, acceptance traces, route ownership, and capture location. Until then, the expected pytest result remains an expectation only.

Blocker B-4 preserves the separately recorded environment-specific reader observation: `journey.resolve-reader-journey-harness-decision` used `python3 tests/check_documentation_smoke.py`, exited `1`, and observed `Operation not permitted` because the evaluator environment denied localhost socket access. The follow-up evaluator owns this unresolved evidence. It does not establish a recurring product harness defect, retry count, affected claim, or repair target; those facts remain unresolved and must not be inferred.

The receipt author and stage worker own only the bounded page, manifest, and registry synchronization. They may compare inputs, preserve exact supplied observations, and route gaps, but they cannot waive a contract mismatch, invent a missing owner, close platform evidence, or convert a validator pass into approval. Candidate authoring feasibility therefore remains blocked until the bounded owners supply authoritative reconciliation or missing evidence.

Read-Only Boundary

This page is a product-side feasibility record. It does not claim platform execution, evaluator-route execution, validator authority, machine identity, evidence-chain completion, independent review, human approval, activation, promotion, publication, scheduling, rollback, release-pointer creation, or a live-directory effect. Candidate-only means that a possible handoff remains parked and bounded; it does not mean approved, publishable, or ready for a transition.

Only the three declared product outputs are in scope: this page, `journeys/contract-derived-candidate-evidence-preflight-receipt.json`, and its matching `registries/pages.json` entry. The receipt does not edit authority, source records, the candidate-handoff page, the glossary, smoke inputs, tests, generated projections, releases, publication pointers, schedules, rollback targets, or a live directory. Generated output remains derived and cannot become semantic source through this receipt.

If a later governed evaluator supplies reader evidence, that record must keep the journey ID, natural-language goal, exact allowlisted command, exit status, observed output, source references, acceptance traces, route owner, and any recurrence facts. An environment or permission failure must remain an evidence blocker until its owning authority establishes product impact. This receipt cannot infer a defect, repair target, approval, or transition from an absent capture, an expectation, or a supplied deterministic validator result.

Acceptance Checks

- **AC-1 — Canonical authority and inputs:** The reader identifies authority revision `2026-08-08`, the exact three logical source IDs, matching public-safe registry records, `employee_may_modify: false`, and the declared contract, guidance, smoke, validator, and output inputs. Private governance material, missing provenance, absent artifacts, and unregistered sources remain outside the usable evidence boundary.
- **AC-2 — Coverage and deterministic command:** The reader parses this manifest, confirms exactly AC-1 through AC-3, verifies that required non-exploratory natural-language journeys trace to every criterion, and checks the exact `python3 -m pytest` prefix, exact pytest command, and expected exit code `0`. Preserved validator evidence stays distinct from a reader journey that this authoring step did not claim to execute.
- **AC-3 — Blockers, ownership, and boundary:** The reader records B-1 through B-4, routes each to its bounded owner while retaining unknown identities as unresolved, separates product evidence from platform attestations, and confirms that no approval, activation, promotion, publication, scheduling, rollback, release-pointer, or live-directory action follows from the receipt.

These checks establish only a synchronized, read-only authoring contract. They do not resolve the acceptance divergence, supply the glossary companions, create reader execution evidence, or classify the localhost observation as a product defect. Those outcomes require authoritative follow-up from the owners named above.

Cross-Repository Start-Here Map for Governed Execution Systems

Audience: onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Cross-Repository Start-Here Map for Governed Execution Systems

This is the public-safe Employee Onboarding entry point for a new human engineer or AI employee with no repository context. It maps the governed documentation product workspace and the named cross-repository references without treating a reference as permission. The intended first outcome is narrow: locate the sealed source boundary, choose the product workspace for a bounded documentation task, and leave unsupported external details unresolved rather than inferring them. Product evidence from this route remains candidate-only. By naming the correct product workspace, bounded repository roles, first-task sequence, and stop conditions, the map is intended to reduce avoidable clarification requests for new engineers without inventing external details.

Candidate-only boundary: this work does not approve, promote, publish, schedule, roll back, create a pointer, write a live directory, or grant platform authority.

Start Here

Read the `Start Here` section of `content/public/engine-overview.md`, then open `sources/authority.json` and resolve its three IDs in `registries/sources.json`. Read `architecture.md` for the product workspace roles before using the journey contract in `journeys/cross-repository-start-here-map.json`. The authority file is a sealed input: this page may describe its public-safe references, but it does not widen, edit, or replace them.

Repository Roles

The current repository, `governed-execution-engine-docs`, is the Documentation Steward product workspace identified by the project onboarding notes. It is the correct repository for this page, its page-registry record, its reader contract, and the deterministic projection. Its `content/` directory is the authored semantic source; `registries/` records identity and references; `models/` contains product architecture data; and `build/index.json` is generated output. The named repository `ai-employee-documentation-steward` appears in the sealed authority only as the separate governing repository for employee and platform controls. Treat it as read-only subject matter here: the product workspace cannot edit or reproduce its private governance.

The three sealed logical source records are the only cross-repository detail this map relies on. `source.naming-decision` resolves to the `factory:` naming locator, `source.documentation-boundary` resolves to the `product:` architecture locator, and `source.platform-delivery-contract` resolves to the `agent-orch:` delivery-contract locator. Their registry records carry revisions, digests, and `public-safe` disclosure. A locator prefix does not establish a remote repository layout, owner, API, or capability. Any external detail not present in those sealed records remains unresolved rather than inferred.

Authority and Action Matrix

The action matrix is a product-side boundary for onboarding. `sources/authority.json` lists exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and records `employee_may_modify: false`. That fact prevents this map from presenting a product task as authority over the governing repository. Read-only inspection of the sealed records is permitted; unsupported conclusions, source expansion, and control changes are not. The local release boundary is also separate: a generated projection or candidate package does not become approval, publication, or platform evidence.

Boundary	Permitted bounded action	Not authorized by this map
Sealed authority and source registry	Read the three IDs, compare their recorded locator, revision, digest, and `public-safe` disclosure, and preserve a gap.	Edit authority, add a source, change disclosure, or infer missing provenance.
Product workspace	Author this public-safe page, synchronize its one registry entry, describe the reader contract, and run the deterministic projection.	Modify the separate governing repository or platform controls.
Read-only subject matter	Use the named governing repository and source locators only for the roles explicitly recorded in sealed references.	Copy private governance, reconstruct unsealed details, or treat a locator as permission.
Release boundary	Keep the result as reversible candidate-only product evidence.	Approve, promote, publish, schedule, roll back, create a pointer, activate, or write a live directory.

If a source ID, revision, digest, disclosure, repository role, or external fact is missing, stale, or conflicting, record that condition as unresolved. Do not repair the gap with plausible prose, a guessed repository, or a new authority claim. The matrix applies equally to a human engineer and to an AI employee.

Safe First Task

For this slice, select `governed-execution-engine-docs` as the work location. The safe bounded first task is to start from the public overview, verify the three sealed source IDs against the public-safe source registry, read the product architecture, and inspect the preserved pinned journey contract. If a concrete map-package defect is identified, this step may repair only this map, its single `registries/pages.json` identity record, and this preflight. The pinned journey manifest and any generated projection remain preserved boundaries; they are not repair targets in this step. This choice is based on the product workspace role and the explicit artifact boundary, not on an inferred capability of any external repository. The named governing repository and every `factory:`, `product:`, or `agent-orch:` locator are read-only context for this task.

The task ends after the reader can explain where semantic content, registry identity, models, generated projection, and release boundaries live. A reader must not begin by changing `sources/authority.json`, `registries/sources.json`, platform controls, release records, publication pointers, or a live directory. If the correct workspace or a needed source fact cannot be established from the sealed inputs, stop and report the unresolved detail. No productivity, readiness, activation, approval, customer, or platform outcome is implied by completing this orientation.

Preflight Receipt

This receipt records product preparation for Employee Onboarding. Its declared inputs for this step include `AGENTS.md`, the sealed authority artifact, `sources/authority.json`, `registries/sources.json`, `architecture.md`, `docs/project-onboarding.md`, `content/public/engine-overview.md`, this map, its `registries/pages.json` identity record, the pinned journey contract, and this preflight. The journey supplies ordered reader checks and exact allowlisted commands, but it is preserved and is not a repair target here. The deterministic projection is generated from semantic source; it is not hand-authored evidence. No trusted artifact inputs are declared for this step.

The preserved independent system-validator record is authoritative evidence from the preceding authority-seal step, not a command run by this authoring step. Its command array is `python3 tools/validate_content.py` and its declared shell command is `PYTHONDONTWRITEBYTECODE=1 python3 tools/validate_content.py`. The record reports exit status `0`, `passed: true`, duration `0.495559` seconds, and evidence hash `d4abd0c9b8c1f4f861d2b154537e2c551c8bd3810232df1c4aa84349781b37c3`. Its stdout is the seven recorded pass lines: `PASS: schema and references`, `PASS: model and craft schemas`, `PASS: internal links`, `PASS: protected and generated paths`, `PASS: reader journeys`, `PASS: public safety`, and `PASS: reproducible projection`; stderr is empty. Validator rerun is not permitted for this step, so this receipt preserves that record rather than replacing it with a worker assertion.

The reader must be able to make these literal acceptance checks:

- **AC-1:** The reader locates the product workspace and the named governing repository through the three sealed source references, explains their bounded roles, and keeps unsupported external details unresolved.
- **AC-2:** The reader selects the product workspace for a safe bounded first task, distinguishes read-only subject matter from product sources, and follows the authority-and-action matrix without changing controls.

- **AC-3:** The reader preserves `employee_may_modify: false` and the candidate-only stop condition, treating product checks as evidence only and not as approval, publication, promotion, rollback, scheduling, pointer, live-directory, or platform authority.

The receipt closes at a candidate-only handoff. When concrete map-package feedback exists, the bounded repair route may touch only `content/cross-repository-start-here-map.md`, `registries/pages.json`, and `docs/cross-repository-start-here-map-preflight.md`. It preserves the pinned journey manifest, authority and source registries, candidate-only boundary, and every unrelated product artifact. It does not authorize an employee cycle, route change, governance edit, release transition, or live publication. A missing or conflicting external detail remains unresolved until an authoritative, public-safe source supplies it.

Customer Capability Inventory

Audience: prospective-client, customer-administrator, adoption-lead | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Customer Capability Inventory

This public page is a bounded inventory of the three source records allowed by the sealed authority. The seal establishes provenance and public-safe disclosure metadata only. It establishes no customer capability, customer outcome, adoption result, documentation-surface description, or decision- support conclusion. Those facts remain unresolved unless a later authority explicitly establishes them.

Customer Capability Gap

The sealed authority establishes no customer capability claim and no customer outcome. Its inventory boundary says that the authority and source metadata establish provenance and disclosure rules only. It does not establish a documentation surface, semantic source, evidence model, reading mode, evaluation route, feature, implementation, benefit, or customer result. Each such description is unresolved and must not be inferred from the registered records. No additional capability or customer outcome is established.

Approved Source and Disclosure

The approved public disclosure is limited to registered reference metadata: locator, revision, digest, authority class, and disclosure. The three authority-listed source IDs below are the complete approved set. The table does not summarize source contents or derive a capability or outcome. A missing, stale, conflicting, or non-public-safe source fact remains unresolved and is not replaced with inferred prose.

Source ID	Locator	Revision	Digest
`source.naming-decision`	`factory:decisions.md#Governed Execution Documentation Steward naming and route`	`2026-08-02`	`7c42ccc9f454a81e2c119c14cdc3ad5ba30ef54d27fb6a8b3`
`source.documentation-boundary`	`product:architecture.md#Semantic source`	`2026-08-04`	`0094e6f534a04729008a3c85510899e5022cbd18a0f7ac930`
`source.platform-delivery-contract`	`agent-orch:docs/delivery-profile-contract.md#content`	`agent-orch-main-e65a64fcf429`	`d466a6dca8420477f416a535c9c2bd76aa314fb5e67d72d40`

Observable Adoption Result

No observable adoption result is established. The sealed inventory lists no established customer capability claims and no established customer outcomes, and it provides no customer activity or result to report. Accordingly, this page cannot report a completed review, use-case decision, adoption, benefit, or customer result. Any statement that a customer completed a review or received an outcome is unresolved and is not supported by this metadata.

Acceptance Checks

1. **AC-1 — Metadata-only boundary:** The reader can state that this inventory exposes approved reference metadata only and establishes no customer capability, customer outcome, adoption result, or decision-support claim.
2. **AC-2 — Unresolved descriptions:** The reader can identify documentation-surface, semantic-source, evidence-model, reading-mode, evaluation-route, and decision-support descriptions as unresolved rather than established by the sealed source-revision metadata.
3. **AC-3 — Approved provenance:** The reader can trace the page to exactly the three authority-listed source IDs and match each registered locator, revision, digest, authority class, and disclosure without inference.
4. **AC-4 — Disclosure boundary:** The reader can state that all three source records are public-safe, that only permitted reference fields appear, and that missing, stale, conflicting, or non-public-safe facts remain unresolved.
5. **AC-5 — No adoption result:** The reader can state that the sealed inventory establishes no customer capability claim or customer outcome and therefore reports no completed review, adoption, benefit, or customer result.
6. **AC-6 — Synchronization:** The reader can verify that the page metadata, pages registry entry, and journey manifest identify the same page and source references while treating unsupported product conclusions as unresolved.

Evidence Artifact Glossary

Audience: new governed-workflow contributors, future Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Evidence Artifact Glossary

This glossary is a bounded aid for new governed-workflow contributors joining implementation or review. It explains how to read the evidence for one first task without turning a product observation into a platform attestation. The authority seal and its read-only inputs define the evidence boundary: the authority revision is `2026-08-08`, the three logical source IDs are `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and `employee_may_modify` is `false`. Only matching public-safe source metadata and conclusions directly supported by the recorded evidence belong in this page. A missing, stale, conflicting, or private input stays unresolved.

Term	Meaning in this bounded first task
Evidence artifact	An inspectable record used to support a product-side conclusion. It retains its location or command, observed result, source references, and acceptance trace instead of relying on plausible prose.
Authority artifact	The sealed authority record and its matching source-registry records. They bound which source IDs and public-safe metadata may be used; they are read-only inputs, not a new grant of employee or platform authority.
Validation artifact	The recorded command, exit status, and output from a deterministic product check. It can support a product conclusion about the documentation route, but it does not establish independent review, approval, activation, or publication.
Smoke artifact	A command-level observation from a smoke check. The known observation here is the single reader-gate attempt whose exact command was `python3 tests/check_documentation_smoke.py`; its exit status was `1` and its output was `Operation not permitted` because localhost socket access was denied by the evaluator environment.
Reader-test artifact	A reader-evidence record for a journey. It should preserve the journey ID, natural-language goal, exact command, exit status, observed result, source references, and acceptance traces. Reader evaluation remains distinct from the source authority and from platform-owned route or execution attestations.
Review artifact	An independent review record and conclusion. It is separate from validation, smoke, and reader-test evidence. This bounded input does not establish a fresh review result, so one must not be claimed from the other artifacts.
Product conclusion	The narrow conclusion supported by product evidence, such as classifying the one recorded observation as an environment-specific evaluation blocker. It does not enlarge the source boundary or speak for platform-owned controls.
Platform-owned attestation	Evidence about route, identity, validator authority, execution environment, evidence chain, approval, activation, promotion, publication, scheduling, rollback, a release pointer, or a live directory. The glossary may distinguish these boundaries but cannot create or infer their attestations.
Candidate-only	A parked product handoff that remains available for review. It is not approval, promotion, publication, scheduling, rollback, activation, pointer creation, or a live-directory write.
Environment-specific evaluation blocker	The status supported by the one denied localhost-socket attempt. It explains the observed evaluation failure without establishing a recurring product harness defect, an affected claim, or a repair target.
Repair target	A named product item that may be proposed only after authoritative follow-up supplies the journey ID, exact allowlisted command, exit status, observed output, retry or recurrence evidence, source references, and affected work item. None is established here.

Using the Glossary in Your First Governed Task

Use the following path for a bounded evidence review. Start with `artifacts/glossary-authoring/authority-seal.json`, then inspect `sources/authority.json` and match every listed ID to its record in `registries/sources.json`. Check the authority revision, `employee_may_modify`, authority class, locator, revision, digest, and `public-safe` disclosure. The separate governing repository is represented only by digest-backed references; its private contents are not a source for this page.

Next read `models/evidence-artifact-glossary-contract.md` and `artifacts/glossary-authoring/evidence-artifact-glossary-journeys.json`. The required product-side set is the authority seal, the authority record, matching public-safe registry records, the contract, the synchronized journey manifest, and a later reader-evidence record with the required fields. Compare this page's metadata with its one matching record in `registries/pages.json`: the page ID, path, title, visibility, audiences, reading modes, claim references, and source references must agree. The Markdown and registry are authored product sources; `build/index.json` is generated projection output and is not authority or a substitute for either source.

For the recorded reader-gate observation, preserve exactly `journey.resolve-reader-journey-harness-decision`, `python3 tests/check_documentation_smoke.py`, exit status `1`, and `Operation not permitted`, with the context that the evaluator environment denied localhost socket access. Treat that as an environment-specific evaluation blocker. The single attempt supplies no recurrence or retry evidence, affected claim, product defect, or repair target. Keep the issue unresolved and route only a permitted follow-up evaluation in an environment with the required localhost reader service and socket access. A later record must retain the journey ID, exact allowlisted command, exit status, observed output, retry count or recurrence evidence, source references, and affected work item before a repair target can be treated as established.

The preserved Agent-Orch validator evidence supplied for this handoff is authoritative captured evidence and is not replaced or rerun by this glossary. Its two recorded product-contract checks both exited `0` and passed with ten tests passed, zero failed, and zero skipped; those results do not become platform attestations. A candidate-only handoff may preserve this evidence and the unresolved reader result, but it cannot create a release pointer, promote, publish, schedule, roll back, activate, or write a live directory.

Sources and Term Boundaries

The glossary exposes only these registered logical sources, all marked `public-safe` in `registries/sources.json`:

Source ID	Authority class	Locator and revision	Digest
`source.naming-decision`	`canonical-platform-source`	`factory:decisions.md#Governed Execution Documentation Steward naming and route`; `2026-08-02`	`7c42ccc9f454a81e2c119c14cdc3ad5ba30ef54d27fb6a8b371`
`source.documentation-boundary`	`product-owned-public-safe-source`	`product:architecture.md#Semantic source`; `2026-08-04`	`0094e6f534a04729008a3c85510899e5022cbd18a0f7ac9302b`
`source.platform-delivery-contract`	`canonical-platform-source`	`agent-orch:docs/delivery-profile-contract.md#content`; `agent-orch-main-e65a64fcf429`	`d466a6dca8420477f416a535c9c2bd76aa314fb5e67d72d405b`

These records provide provenance metadata and a public-safe disclosure boundary; they do not expose private source corpora or the contents of the separate governing repository. If any ID, revision, digest, authority class, locator, or disclosure does not match, record the gap as unresolved and make no inference. In particular, do not invent a source map, claim, harness defect, affected work item, approval, or repair from generated output.

Keep the artifact roles separate. Authority selects the permitted source boundary. Validation records a deterministic product check. Smoke records a command-level observation. A reader-test record describes a reader journey and its exact evidence fields. A review artifact records an independent review conclusion. Product evidence can support a bounded product conclusion, while platform-owned attestations remain with their owners. The candidate-only default is reversible: retain the unresolved handoff and request permitted follow-up evidence. No term in this glossary changes governance, routing, validator ownership, activation rules, publication policy, release-pointer state, rollback authority, or the live-directory boundary.

Evidence Handoff Preflight and Template

Audience: agent-orch maintainers preparing an operator-facing candidate evidence handoff, Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Evidence Handoff Preflight and Template

Purpose

The primary workstream for this preflight is **Internal Knowledge**. Its audience is agent-orch maintainers preparing an operator-facing candidate evidence handoff, Internal Knowledge playbook authors who will reuse the format, and independent readers who need to inspect the result. The intended outcome is a bounded, readable handoff that explains the reader problem, recommendation, acceptance coverage, evidence ownership, and unresolved facts well enough for a later checklist author to work from the same inputs. This is product-side documentation guidance: it does not change sealed authority, create a platform attestation, or authorize approval, activation, promotion, publication, scheduling, rollback, a release pointer, or a live directory.

The preflight is deliberately earlier than checklist authoring. It turns the proposed handoff into an evidence inventory before prose can make an unsupported conclusion look complete. A maintainer should be able to say which workstream was selected, what outcome the reader needs, which sources own each fact, what smoke and journey records exist, which deterministic commands can be re-executed, and which version or rollback facts remain unresolved. Missing evidence stays visible instead of being filled with plausible wording.

Preflight Procedure

Begin by recording the selected workstream, audience, reader problem, and intended outcome. Then compare the proposed acceptance criteria with the reader-journey manifest: every criterion must have a concrete artifact, an owning authority or explicitly unresolved owner, and a non-exploratory journey trace. If acceptance coverage is missing, name the missing criterion and stop before drafting a checklist; a checklist cannot repair absent evidence. Keep product conclusions separate from platform-owned validator, route, identity, terminal-state, approval, or publication attestations.

Next inspect the smoke artifacts. Confirm that the named smoke manifest, checker, input page, registry record, and captured output location exist or are explicitly marked unresolved. Record the exact exit status and observed output for each supplied attempt, including a permission or environment blocker when that is what the reader observed. A missing smoke artifact is a preflight finding, not a reason to infer that the smoke passed or that a product defect exists. Preserve failed evidence beside any later result.

Finally, check re-executability before checklist authoring. Every deterministic command must be a non-empty string in the selected manifest's `command_allowlist`, resolve to the declared repository inputs, and identify where its output will be recorded. Reject a stale, substituted, or unallowlisted invocation and record the mismatch. Compare the page metadata with its single `registries/pages.json` entry and treat `build/index.json` as generated output. Only when acceptance coverage, smoke artifacts, and exact commands are accounted for should the maintainer draft a checklist and assign a review verdict. The handoff remains candidate-only throughout.

Candidate Evidence Handoff Template

Copy the following template into a candidate record and replace every placeholder with an observed value or the explicit word `unresolved`. Keep the labels stable so an independent reader can compare handoffs. “Authority/source evidence” identifies who owns a fact and where its supporting record lives; it is not a claim that a platform gate ran. “Smoke evidence” and “user-journey evidence” retain exact observations, including failures. “Previous-version identity” and “rollback-target facts” must be copied from the candidate evidence when present and must remain unresolved when no authoritative record establishes them.

```
candidate_evidence_handoff:
  selected_workstream: "Internal Knowledge"
  audience: "<operator-facing maintainers, playbook authors, and/or independent readers>"
  intended_outcome: "<bounded reader outcome>"
  reader_problem: "<specific problem the reader needs this handoff to solve>"
  recommendation: "<bounded recommendation, or unresolved>"
  acceptance_criteria:
    - id: AC-1
      statement: "<criterion and its concrete artifact>"
      evidence_location: "<path or stable evidence id>"
      journey_refs: ["<journey id>"]
    - id: AC-2
      statement: "<criterion and its concrete artifact>"
      evidence_location: "<path or stable evidence id>"
      journey_refs: ["<journey id>"]
    - id: AC-3
      statement: "<criterion and its concrete artifact>"
      evidence_location: "<path or stable evidence id>"
      journey_refs: ["<journey id>"]
  authority_source_evidence:
    authority_owner: "<authority boundary or unresolved>"
    source_refs: ["<registered source or evidence path>"]
    observed_result: "<revision, digest, disclosure, and conclusion>"
  smoke_evidence:
    manifest: "<smoke manifest path>"
    checker: "<checker path>"
    command: "<exact observed or allowlisted command>"
    exit_status: "<integer or unresolved>"
    observed_output: "<verbatim-short observation or unresolved>"
    output_location: "<captured result path or unresolved>"
  user_journey_evidence:
    manifest: "<reader-journey manifest path>"
    journeys: ["<journey id>"]
    traces_to: ["AC-1", "AC-2", "AC-3"]
    observed_result: "<reader conclusion, including retained blockers>"
  deterministic_commands:
    - command: "<exact non-empty command from command_allowlist>"
      purpose: "<bounded check>"
      output_location: "<result path or unresolved>"
      exit_status: "<integer or unresolved>"
  review_verdict:
    value: "<pass, findings, blocked, or unresolved>"
    reviewer_evidence: "<independent review record or unresolved>"
  candidate_only_publication_boundary: "<state that this is candidate-only and no approval, promotion, publication, s
  previous_version_identity: "<exact prior version identifier, or unresolved>"
  rollback_target_facts:
    target_identity: "<exact rollback target, or unresolved>"
    available: "<true, false, or unresolved>"
    evidence_location: "<candidate artifact path or unresolved>"
```

After copying, check that every placeholder is resolved or intentionally marked unresolved, each command is exact, and each acceptance criterion points to evidence and a journey. A review verdict describes the product evidence; it does not establish platform execution or human approval. Leave the candidate parked if source authority, smoke output, journey coverage, prior-version identity, rollback target, or any other required fact is missing. The template is complete only when a reader can tell what was observed, what remains open, and who owns the next evidence-producing action.

Acceptance Checks

- **AC-1 — Scope and coverage:** The reader confirms that **Internal Knowledge** is the selected workstream, the named audience and intended outcome are explicit, and the reader problem and recommendation are bounded. Before checklist authoring, the reader maps AC-1, AC-2, and AC-3 to concrete artifacts and non-exploratory journey traces; missing coverage is reported as an unresolved preflight finding rather than hidden in checklist prose.
- **AC-2 — Evidence inventory:** The reader verifies that the handoff records authority/source evidence, smoke evidence, and user-journey evidence with source locations, observed results, ownership, and blockers. Missing smoke artifacts, failed attempts, or platform-owned attestations are retained as facts about the evidence boundary; they are not rewritten as a passing product conclusion. The reader can complete the template without inventing provenance.
- **AC-3 — Re-execution and boundary:** The reader verifies that each deterministic command is exact, non-empty, and present in the selected manifest's `command_allowlist`, with an output location and observed status. The review verdict, candidate-only publication boundary, previous-version identity, and rollback-target facts are recorded separately, with absent facts marked unresolved. The journey ends without inferring approval, promotion, publication, scheduling, rollback, a pointer, or a live directory.

The synchronized reader manifest maps one non-exploratory journey to each check: the maintainer verifies scope and coverage, the playbook author fills the evidence inventory, and the independent reader verifies command re-execution and the candidate-only boundary. These traces make the preflight auditable before checklist authoring while keeping the handoff within the declared product paths.

Failure-Triage Decision Guide

Audience: auto-orch operators, workflow maintainers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Failure-Triage Decision Guide

Purpose

This is an Internal Knowledge guide for auto-orch operators and workflow maintainers. The reader outcome is to consistently classify strict-lint and evidence-chain failures, route each to the correct owner, reject unsupported workarounds, and preserve complete escalation evidence. The sealed authority map records revision `2026-08-08`, exactly three logical source IDs, and `employee_may_modify: false`; each registered source is public-safe. Verify those metadata facts before classifying an observation, but treat them as a bounded source boundary rather than as evidence of a runtime event or a cause. They do not expose the referenced source corpora or runtime evidence, and they do not grant authority to change governance, routing, validators, activation, approval, publication, promotion, rollback, scheduling, or a live directory.

Classify the Failure

Start with the observed record and its exact location, not with a guessed cause. First verify that `sources/authority.json` and `registries/sources.json` still identify the sealed set: `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, with public-safe disclosure. Use the strict-lint label only when the supplied observation identifies a deterministic product-input problem such as page structure, metadata, a registered reference, or another declared content check. Record the failing path, rule, and actual result; do not turn a missing observation into a lint failure.

Use the evidence-chain label when the supplied record concerns the linkage or completeness of an escalation record, such as its source mapping, command claim, status, observed output, or acceptance trace. This is a record-quality classification, not proof of a product defect. Keep it separate from an environment or platform fact: runtime behavior and platform attestations are not established by a product-side record. An absent, conflicting, or unsealed item remains unresolved until its owning boundary supplies it. A category describes only the observed gap; it is not permission to infer a cause, repair target, owner, retry, or terminal outcome.

Route the Owner

Route a strict-lint finding to the product-side content or workflow maintainer when the evidence points to a Markdown page, page metadata, registry entry, source reference, or deterministic projection input. The maintainer may repair the declared product source and retain the original observation, but may not rewrite the sealed authority or source registry to make the result pass. Route an evidence-chain finding about runtime execution, evaluator evidence, validator authority, routing, identity, or other platform attestation to the owning evaluator or platform boundary. These are separate routes: do not convert a product-content defect into a platform attestation, or a missing platform attestation into a product defect. A workflow maintainer may package the product-side handoff and reconcile its references; that does not certify the external evidence.

The sealed map names ownership boundaries, not a person, queue, or new routing authority. If the supplied evidence names no owner, record exactly `owner unresolved`, preserve the evidence, and escalate through the governed route already responsible for that boundary. Do not choose an owner from a filename, a generated projection, an inferred source, or an unregistered runtime claim. A successful local check can support a product conclusion, but cannot be used as a platform attestation or as permission to cross a separate control boundary.

Preserve Escalation Evidence

Keep the original observation beside every later result; never replace the first failure with a later pass. The escalation record should name the journey or work item, the exact non-empty command claimed by the manifest, the exit status, the observed output or mismatch, the relevant page or source path, the source references, and the `traces_to` acceptance checks. For this guide the declared command is exactly `PYTHONDONTWRITEBYTECODE=1 python3 -m`

`pytest`; its presence in an allowlist is only a re-executable product claim, not proof that a platform executed it. Record whether each observation is strict-lint, evidence-chain, or unresolved, and identify the owner boundary without filling unknown fields with prose.

Preserve the sealed authority revision, the three logical source IDs, their public-safe status, and the `employee_may_modify` value as metadata. Do not copy private governance material, private source corpora, or runtime evidence into the page. Reject an unsupported workaround, including changing a source, claim, command, or authority record merely to silence a mismatch. Do not hand-edit `build/`, or treat a missing attestation as a failure owned by product content. A complete escalation makes the gap auditable and reversible while leaving publication, promotion, scheduling, rollback, and live-directory state untouched.

Acceptance Checks

- **AC-1 — Classify from sealed inputs:** The reader verifies authority revision `2026-08-08`, the exact three logical source IDs, public-safe registration, and `employee_may_modify: false`; then the reader labels only an observed deterministic product-input mismatch as strict-lint and leaves missing, stale, conflicting, or runtime-only facts unresolved.
- **AC-2 — Route by ownership boundary:** The reader routes a product-side strict-lint finding to the content or workflow maintainer, routes runtime or platform-attestation evidence to its owning evaluator or platform boundary, and records `owner unresolved` when the supplied evidence names no owner; the reader does not invent a person, queue, or new authority.
- **AC-3 — Preserve a complete escalation:** The reader retains the original failure beside later results with the journey or work item, exact command, status, observed output, paths, source references, and `traces_to` values, rejects unsupported workarounds, and distinguishes product conclusions from platform-owned execution or validator attestations.

First governed documentation contribution

Audience: New governed-documentation contributors | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

First governed documentation contribution

Your First Safe Contribution

Treat the first contribution as a small, inspectable product change. Start at the repository root, read the onboarding notes and the sealed authority, and choose one documentation improvement that can be explained in one sentence. The safe result is a candidate page plus the records that let another reader trace its identity, sources, journey goals, and validation. It is not a claim that the contributor has approval, platform access, readiness, or a customer outcome.

Keep the contribution narrow enough that a reviewer can compare every material sentence with the three authority-backed source records. Do not fill a missing fact with plausible prose. If the source boundary, page identity, or intended audience is unclear, preserve the uncertainty in the candidate record and ask for the separately governed decision. A useful first task demonstrates reproducible product evidence and a clean handoff, rather than maximizing the amount of text changed.

Repository Roles

This `governed-execution-engine-docs` repository is the Documentation Steward product workspace. `content/` holds semantic Markdown, `registries/pages.json` holds page identity and provenance metadata, and the `journeys/` record holds natural-language reader goals and acceptance traces. The candidate record in `releases/` describes the parked product handoff. `build/index.json` is a deterministic projection of product sources, so it is an output to inspect and never a source to edit.

Select the documentation-steward author role for this task: orient, author one bounded page, synchronize its registry entry, record reader goals, generate the projection, and stop at independent review. The separately governed employee repository and its employee, routing, validator, activation, approval, and publication controls are authority paths, not product files to edit here. The names of external repositories do not grant access or permission, and no unsupported details about their layout or operators should be inferred.

Authority Boundaries

Open `sources/authority.json` as a sealed, read-only input. It has revision `2026-08-08`, sets `employee_may_modify` to `false`, and lists exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`. Match each logical ID in `registries/sources.json`, confirming its locator, revision, digest, authority class, and `public-safe` disclosure before using it in the page header or registry. The page uses no claim references because no registered product claim is needed for this orientation guide.

The editable product records for this task are the new Markdown page, its one `registries/pages.json` entry, the candidate record, and the journey manifest. The authority file and source registry are read-only evidence; the governing repository is outside this workspace; and `build/` is generated. A missing, stale, conflicting, or non-public-safe source is an unresolved boundary, not a reason to edit authority, copy private governance material, invent a claim, or declare the task complete.

Candidate-Only Workflow

First choose the page path and stable ID, then write the metadata block with the exact title, public-safe visibility, audience, reading modes, empty claim references, and the three sealed source references. Add the same identity and reference fields to `registries/pages.json`. Next create the four reader journeys in `journeys/first-task-guide-user-journeys.json`: each journey should state a natural-language goal, name its author authority, remain non-exploratory, and trace one of AC-1 through AC-4. Keep the candidate release record synchronized with the page, workstream, evidence locations, and candidate-only boundary.

Run the declared deterministic product path from the repository root: `python3 tools/validate_content.py`, followed by `python3 tools/build_projection.py`. Inspect `build/index.json` for the new page ID, path, title, visibility, claim references, source references, content revision, and authority identity. Fix semantic sources when a check exposes a gap; do not patch the generated projection. The candidate remains parked for review even when these product checks pass, and no release pointer, live directory, promotion, publication, schedule, rollback, or activation follows from this workflow.

Review Handoff

Hand the candidate package to an independent reader with the Markdown page, the synchronized page registry entry, the journey manifest, the sealed source references, the candidate record, and the generated projection location. Ask the reader to follow each natural-language goal and preserve the exact command, exit status, observed output, and any unresolved blocker. A declared command in a manifest is an allowlist declaration, not evidence that a platform ran it; the author must not turn an expected result into an attestation.

After the reader record is available, route the package to an independent reviewer on a separate review route. The reviewer checks scope, provenance, heading and metadata requirements, page-registry synchronization, journey coverage, and the candidate-only boundary. Keep the reviewer’s verdict and findings separate from the author’s summary. If reader or review evidence is missing or fails, retain that fact and leave the candidate parked; do not narrow the journey, delete the failed evidence, self-certify a pass, or publish.

Acceptance Checks

- **AC-1 — Repository orientation:** The contributor identifies this repository as the Documentation Steward product workspace, explains the roles of `content/`, `registries/`, `journeys/`, `releases/`, and generated `build/`, selects the documentation-steward author role, and leaves unsupported details about the separately governed repository unresolved.
- **AC-2 — Authority boundaries:** The contributor reads the sealed authority, verifies revision `2026-08-08`, `employee_may_modify: false`, and the exact three public-safe source IDs, then uses only those IDs for page provenance without changing authority or copying private governance material.
- **AC-3 — Compliant candidate creation:** The contributor creates one narrow candidate-only page with synchronized identity fields, the named audience, appropriate reading modes, only authority-backed references, four traced journey goals, and a deterministic projection generated from source files.
- **AC-4 — Review handoff:** The contributor packages the candidate for an independent reader and an independent reviewer, keeps product conclusions separate from platform attestations, preserves missing or failed evidence, and confirms that no approval, promotion, publication, scheduling, rollback, release-pointer, activation, or live-directory action occurred.

First-Task Guide

Audience: onboarding-teammate, new governed-documentation contributors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

First-Task Guide

This is a practical first task for a new auto-orch operator working on a candidate-evidence handoff. It is a product-side walkthrough: inspect the declared contract, run the canonical read-only preflight against the supplied sample, preserve the exact result, and stop at the candidate-only boundary. The preflight does not grant authority, approve a candidate, or prove a platform-owned transition.

Candidate Evidence Chain Walkthrough

Follow the evidence chain in order. First locate the human-readable contract at `docs/candidate-evidence-preflight-contract.md`, then compare its machine-readable candidate-handoff declarations in `journeys/candidate-evidence-handoff-manifest.json` and this page's `content/first-task-guide-reader-journeys.json`, alongside the model, executable tool, and sample evidence. The contract explains the required fields and exact command/name matching; `models/candidate_evidence_handoff_contract.json` carries the synchronized shape; `tools/candidate_evidence_preflight.py` checks declarations, supplied observations, and the independently supplied review verdict; and `tests/fixtures/candidate_evidence/valid.json` is the sample candidate to read. Keep declarations (what a journey says should have happened) separate from evidence (what the supplied record says was observed).

The chain ends in a product conclusion: the supplied record either satisfies the read-only preflight or is rejected with discrepancies. The tool does not execute commands claimed inside the evidence and does not write a result artifact. A pass is therefore evidence that this supplied handoff is contract-complete, not evidence of approval, publication, promotion, scheduling, rollback, release-pointer creation, activation, or a live-directory change.

Before You Run the Preflight

Open the contract before opening the sample. Confirm its synchronized machine-readable companion, the model, the preflight tool, and the valid fixture; then note the required independent review path `code-reviews/content-review.verdict.json`. A named review path is a required location, not proof that a review artifact exists. Also keep the sealed authority and registered source boundary in view: `sources/authority.json` and `registries/sources.json` are read-only product inputs, and the guide's three registered source references must not be supplemented with guessed provenance.

Do not edit the valid fixture to make it agree with a declaration, do not rewrite the contract or model while investigating, and do not treat a fixture's `commands_run` entries as commands that this preflight will replay. Capture the input path and the exact command you use. If a declaration and an observed field disagree, retain both values so a later reader can see whether the discrepancy belongs to the manifest, the evidence record, or the review verdict.

Run the Canonical Preflight

From the repository root, run the canonical read-only preflight against the valid sample exactly as follows. Pin the candidate-handoff manifest and contract explicitly so the invocation cannot silently select an unrelated journey manifest:

```
PYTHONDONTWRITEBYTECODE=1 python3 -B tools/candidate_evidence_preflight.py --evidence tests/fixtures/candidate_evider
```

The tool reads the selected manifest, contract, and sample, checks the required journey coverage, exact names, traces, allowlisted commands, integer exit codes, observed-result fields, and review-verdict path, and returns exit code `0` with a JSON `passed: true` and `verdict: "pass"` when all supplied facts agree. A non-JSON invocation reports the equivalent `PASS: candidate-evidence preflight passed` line. If the selected manifest is stale or conflicts with the pinned handoff, the command returns exit code `1` and emits the exact declaration or evidence discrepancies; that rejection is the result to preserve. It

is read-only: it does not run the commands named in the sample, edit the sample, alter authority, change a validator, or create a publication pointer. Preserve the stdout, stderr, exit code, input path, and any machine-readable error list as the run evidence.

The observed attempt for this walkthrough rejected the supplied declarations: it reported `manifest.required_review_verdict_path`, a missing `manifest.evidence_schema`, a missing `manifest.review_verdict`, and missing `required` and `expected_exit_code` fields on all three candidate-handoff journeys. Keep that rejection beside the sample and route it as declaration discrepancy evidence. Do not turn it into a pass by editing authority, the fixture, the validator, or the independent review evidence; a later governed manifest or review result may supplement the record but cannot erase this observation.

Interpret Passing and Failing Results

A passing result means the supplied record is structurally and declaratively consistent with the canonical handoff contract: the pinned journeys are covered, their names and traces match, each claimed command is an exact allowlist member, each claimed exit code has the declared type and value, the observed-result fields are present, and the required review verdict is readable and acceptable. It does not mean that the sample's claimed commands ran during this invocation, that a product conclusion is a platform attestation, or that any transition is authorized.

A rejection means the tool found a discrepancy. Declaration discrepancies can include a changed journey name, missing acceptance coverage, an unallowlisted or whitespace-altered command, a changed expected exit code, or a manifest shape that no longer matches the contract. Evidence discrepancies can include missing observed fields, incomplete journey records, the wrong review path, or a supplied status other than `passed`. Read the exact error and classify it against the original files; do not normalize it away. A rejected sample or conflicting declaration is evidence to preserve and escalate, not a reason to edit authority, fixtures, validators, or review evidence.

Escalate Discrepancies

Escalate with the smallest reproducible packet: the contract and manifest paths, candidate input path, exact command, exit code, stdout and stderr, the tool's exact rejection lines, the affected journey or field, and the source references that establish the expected boundary. State whether the conflict is in a declaration, the supplied candidate evidence, or the independent review record. Keep the original rejected record beside any later corrected or rerun record; a later result supplements history and does not erase the first observation.

Route the discrepancy through the owner named by the governing process. If authority or a registered source conflicts with the candidate, preserve the sealed value and report the conflict for authority review. If the evidence is incomplete, request a new governed evidence record rather than filling fields from assumptions. If the independent review artifact is missing or malformed, leave handoff completion unresolved. Do not repair a failed result by editing authority, fixtures, validators, or review evidence, and do not run an unallowlisted command as an unofficial substitute for the canonical check.

Candidate-Only Boundary

Finish by confirming what did not happen. The preflight read the contract, manifest, candidate sample, and required review path; it did not approve, activate, promote, publish, schedule, roll back, create or change a release pointer, or write a live directory. A passing result remains a product-side candidate-evidence conclusion, and a rejection remains preserved evidence for escalation. Neither result changes the sealed authority or turns a candidate into a release. If a separate deterministic projection is later generated, `build/index.json` is still generated product output, not a publication pointer or a transition attestation.

Before closing the task, record the preflight result and verify that no non-candidate transition occurred: no authority or fixture was rewritten, no validator or review record was altered, no publication boundary was crossed, and no live-directory or release-pointer action was taken. The correct ending is a synchronized, inspectable candidate handoff with its pass or rejection evidence intact and any unresolved discrepancy explicitly escalated.

Governed candidate delivery

Audience: customer-administrator, adoption-lead | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Governed candidate delivery

Governed candidate delivery is a public-safe, bounded capability explainer for customer administrators and adoption leads. It describes how a documented candidate is checked against an approved source boundary before a reader decides whether a narrowly scoped adoption conversation is useful. The page explains evidence and constraints, not a promised customer result, technical performance, security result, price, availability, or implementation outcome. Its references are limited to the three logical sources sealed by the authority manifest. If a source fact is missing, stale, conflicting, or not public-safe, the responsible reader records that unresolved gap instead of turning an assumption into guidance.

How validation and constraints work

Validation starts with the sealed authority manifest, which names the approved logical source IDs. The source registry then supplies the public-safe record for each ID, including its authority class, locator, revision, digest, and disclosure. The page header and its registry entry are compared for one page identity, path, title, visibility, customer audiences, reading modes, claim references, and source references. Deterministic product checks and projection checks can then establish consistency across those product inputs. That result is product evidence about the documented candidate; it is not a platform attestation, an independent review result, an approval, or a publication decision. A constraint or mismatch remains a visible blocker until an authorized source resolves it.

Adopt safely

Begin with one bounded documentation use case and write down the claims and evidence that a customer administrator or adoption lead would need to assess it. Compare those needs with the three approved source records and the facts present in the candidate. Mark every unsupported, missing, stale, or conflicting item as an unresolved gap, and ask for a named authority follow-up when one is needed. The safe next adoption action is a scoped, reversible fit conversation or named-authority follow-up that leaves the candidate parked while the gap is clarified. This action does not promise an outcome and does not perform approval, promotion, publication, scheduling, rollback, pointer creation, or any live-directory write.

Approved sources and limits

The authority manifest identifies exactly three logical sources for this explainer. Their registry records are public-safe and provide the permitted traceability boundary; they do not authorize the page to extend platform authority or to copy private governance material. The source records support the validation flow and the stated delivery limits, but they do not establish customer outcomes or technical capabilities beyond those recorded facts. A reader should stop at the boundary when a source reference cannot be matched or its disclosure is not public-safe.

Source ID	Authority class	Revision	Disclosure
`source.naming-decision`	`canonical-platform-source`	`2026-08-02`	`public-safe`
`source.documentation-boundary`	`product-owned-public-safe-source`	`2026-08-04`	`public-safe`
`source.platform-delivery-contract`	`canonical-platform-source`	`agent-orch-main-e65a64fcf429`	`public-safe`

These are source references, not a new authority grant. The candidate-only boundary remains in force even when the product checks pass: a check can show that the page and its registered references agree, but it cannot create publication authority, a release pointer, a live release, or a live directory.

Frequently asked questions

Is a documented candidate the same as general availability?

No. A documented candidate is a bounded, candidate-only state that remains parked while its approved source boundary and product evidence are examined. Reading this page or passing deterministic product checks does not establish general availability. Those checks can support a product conclusion about the documented candidate, but they are not a platform attestation or independent review and do not create approval, promotion, publication, scheduling, rollback, a release pointer, or a live-directory write. Treat the candidate as candidate-only unless a separately governed authority establishes a different state.

How can I verify that this candidate and its evidence are the right ones?

Start with the approved authority boundary: revision `2026-08-08`, disclosure `public-safe`, and exactly these three source IDs: `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`. Confirm that this page's metadata lists those same three IDs and no others. For each ID, find the matching public-safe record in the source registry and check its authority class, locator, revision, digest, and disclosure. Then check that the page identity, path, title, visibility, audiences, reading modes, claim references, and source references agree with the registered page. Evidence is adequate only for the documented candidate and the bounded checks that were actually performed. A missing, stale, conflicting, unsupported, or non-public-safe record is an unresolved gap; it is not evidence to fill in from assumption.

What should I do after that verification?

Choose one bounded documentation use case and write down the claims and evidence needed to assess it. Compare those needs with the approved source records and candidate facts, and record every missing, stale, conflicting, or unsupported fact as an unresolved gap. Then choose a scoped, reversible fit conversation or a follow-up with the named authority for the missing decision. Keep the candidate parked throughout. This guidance does not establish general availability or authorize a customer outcome, technical performance, approval, promotion, publication, scheduling, rollback, pointer creation, or a live-directory write.

Reader acceptance checks

1. **AC-1 — Validation flow:** The reader can explain the path from sealed authority, through the three approved public-safe source records and synchronized page metadata, to deterministic product evidence, while keeping that evidence separate from platform attestations and publication decisions.
2. **AC-2 — Candidate-only boundary:** The reader can state that constrained delivery stops at a parked candidate and does not become approval, promotion, publication, scheduling, rollback, a release pointer, or a live directory merely because the page or evidence was inspected.
3. **AC-3 — Next adoption action:** The reader can select one bounded use case, record unsupported or unresolved facts as gaps, and choose a scoped, reversible fit conversation or authority follow-up while leaving the candidate parked and making no customer-outcome claim.

Governed Content Cycle Handoff

Audience: Internal operators running governed content cycles | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Governed Content Cycle Handoff

Purpose and Audience

This page is a practical Internal Knowledge handoff for Internal operators running governed content cycles. Use it to carry a bounded workstream choice through source and authority checks, deterministic product validation, reader evidence, independent review, registry synchronization, and a candidate-only transfer. The outcome is traceable product evidence, not a claim of approval, publication, deployment, customer capability, or platform attestation.

Compatibility Inventory

This page is paired with `journeys/governed-content-cycle-handoff-compatibility-inventory.json`, a product-side inventory for Internal operators running governed content cycles. The inventory keeps the page and its reader journey compatible with the sealed inputs without widening the page's authority.

- **Authority boundary:** Read `sources/authority.json` as revision `2026-08-08`. Its exact logical sources are `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`; each matching record in `registries/sources.json` must remain `public-safe`, and `employee_may_modify` remains `false`.
- **Page boundary:** Preserve page identity `page.governed-content-cycle-handoff`, title `Governed Content Cycle Handoff`, public visibility, the Internal operators audience, the three reading modes, an empty claim reference list, and the three sealed source references. No compatibility check creates or changes a claim.
- **Check boundary:** The exact deterministic product command is `python3 tools/validate_content.py`. A declared command is permission to run that check, not execution evidence; retain its exact result separately from reader evidence, review evidence, and platform-owned attestations.
- **Repair boundary:** If a later governed result names a bounded product mismatch, repair only this page or the dedicated inventory manifest. Keep missing or environment-specific evidence unresolved until its owning evidence path supplies it.
- **Transfer boundary:** The result remains candidate-only. This inventory does not approve, promote, publish, schedule, roll back, create or change a release pointer, activate a route, or write a live directory.

Handoff Procedure

1. **Select the internal workstream.** Start with the supplied workstream comparison and select **Internal Knowledge** only when the recorded selection evidence supports it. For every in-scope item, record its queue identifier, observed queue age or age field, age source, review timestamp, and review status. If age is absent, malformed, stale, or conflicting, record the exact gap and do not invent a threshold, priority, or schedule.

2. **Check source gaps and seal authority.** Read `sources/authority.json` as a read-only baseline. The sealed record is revision `2026-08-08`, lists `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and sets `employee_may_modify` to `false`. Match every used logical ID to `registries/sources.json` and confirm its authority class, revision, digest, locator, and `public-safe` disclosure. A missing, stale, conflicting, or non-public-safe mapping is an unresolved source gap, not an invitation to edit the registry or fill the fact with prose.

3. **Run the deterministic product check.** Use only the exact non-empty command declared by the dedicated compatibility inventory: `python3 tools/validate_content.py`. Retain the exact command, exit status, observable stdout or stderr, time if available, and output or artifact location. A command listed in a manifest is permission to run that deterministic check, not execution evidence by itself.

4. **Capture reader evidence and independent review.** Give an independent reader the page, registry, source, contract, and journey inputs named by the manifest. The reader record must identify the journey, exact command, exit status, observed result, source references, and acceptance traces. A missing,

failed, or environment-blocked reader result stays unresolved. Route the completed packet to an approved independent reviewer on a route separate from production; keep reviewer identity, route, verdict, findings, and evidence location separate from the author's conclusion.

5. **Reconcile and transfer.** Check AC-1 through AC-7, reconcile page metadata with its registry entry, inspect the deterministic projection, and retain every unresolved gap. Transfer only the resulting evidence packet to the candidate-only boundary after the required evidence and review records are present or explicitly unresolved.

Use the same receipt shape for every cycle: a selection record with the chosen workstream, in-scope queue identifiers, observed age fields, age sources, review timestamps and statuses; an authority record with the sealed revision, `employee_may_modify` value, logical source IDs, and source-registry mappings; check records with each exact command, exit status, observable output, and artifact location; reader and review records with journey IDs, commands, results, source references, acceptance traces, reviewer route, verdict, and findings; and a reconciliation record with the page metadata, registry entry, projection identity, unresolved gaps, and candidate-only disposition. A field that is absent or conflicting is recorded as unresolved in that receipt.

Current Evidence Boundary

The current platform-owned reader-gate evidence records one attempt for `journey.resolve-reader-journey-harness-decision`, using the exact command `python3 tests/check_documentation_smoke.py`. It exited with status `1` and reported `Operation not permitted` because the evaluator environment denied localhost socket access. This establishes an environment-specific evaluation blocker only; it does not establish a recurring product harness defect, a retry count, an affected claim, or a repair target. Keep the issue unresolved and route a permitted follow-up evaluation.

The selected workstream is **Internal Knowledge**. Until authoritative follow-up evidence supplies recurrence and product impact, affected work is limited to this evidence-routing question, the reader-journeys manifest (`journeys/reader_journeys_manifest.json`), and the candidate-only record. No public semantic page change or product claim change is established by the blocked evaluation.

Evidence and Registry Mapping

Treat the Markdown metadata, `registries/pages.json`, the journey manifest, the sealed authority, the source registry, and `build/index.json` as one traceable handoff. The page identity is `page.governed-content-cycle-handoff`; its path is `content/governed-content-cycle-handoff.md`; its title is `Governed Content Cycle Handoff`; visibility is `public`; its audience is `Internal operators running governed content cycles`; its reading modes are `executive`, `engineer`, and `ai`; its claim references are empty; and its source references are the three sealed logical IDs. The header uses `claims` and the registry uses the corresponding empty `claim_refs`; do not introduce an unsupported claim merely to make a mapping appear complete.

For each source mapping, preserve the registered identifier, authority class, revision, digest, locator, and disclosure. Produce `build/index.json` from the semantic sources and inspect its page identity, title, visibility, claim references, source references, and body digest; never hand-edit generated output. A command listed in the manifest is permission to run that deterministic check, not evidence that it was run. Record any result from the current step separately from reader evidence, independent review, and publication authority.

Candidate-Only Transfer

Transfer only the reconciled packet, its source mapping, deterministic results, reader record, independent review, acceptance matrix, and projection identity to a record explicitly marked **candidate-only**. Candidate-only means that evidence is staged for a separately governed decision; it does not mean approved, active, promoted, published, scheduled, rolled back, or ready for a live directory. If a required result is unavailable, retain the exact unresolved gap and its owner or permitted follow-up instead of asserting a pass.

This handoff strictly prohibits publication or release-pointer creation. Do not create or change `publication/current-release.json`, promote or roll back a release, schedule work, activate a route, alter governing controls, or write a live directory. Do not represent deterministic product validation, reader evidence, or independent review as a platform-owned attestation. The parked candidate remains reversible until the separately authorized authority and evidence path supplies any later transition decision.

Governed Documentation Cycle Lifecycle

Audience: agent-orch maintainers, governed-cycle operators, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Governed Documentation Cycle Lifecycle

Overview

This internal lifecycle map is for agent-orch maintainers, governed-cycle operators, and independent readers working in the **Internal Knowledge** workstream. Its objective is a reproducible path from bounded selection and authority tracing through semantic authoring, deterministic product checks, independent reading, independent review, and a candidate-only handoff. The outcome is a traceable product-evidence packet, not permission to change governance or to take a governed transition.

Bounded Objective

The bounded objective is to let a new operator trace one product-only cycle in order: record the ordered selection of the bounded **Internal Knowledge** slice; seal and read the authority and public-safe sources; author the semantic page and preserve deterministic validation and projection; replay the fixed smoke journey; capture independent reader evaluation; obtain independent review; and make a candidate-only handoff. The cycle ends with a reversible evidence packet and does not establish approval, activation, promotion, publication, scheduling, rollback, a release pointer, a live directory, or a platform attestation.

Read this map with the `lifecycle contract`, the reader-journey manifest, the `sealed authority`, and the `public-safe source registry`. The page identity is registered in `pages.json`; that registration and this page are synchronized semantic sources. The map records public-safe source IDs and reference metadata only. It does not reproduce private governance material, runtime evidence, or an engine-owned attestation.

Operator Problem

A fresh operator must answer six questions in order: what workstream and scope are selected; which authority and source records may be read; what product content is authored and what deterministic result was actually observed; what an independent reader concluded; whether independent review used a separate route; and what bounded packet may be handed off. A selected workstream is not transition authority. A command in the manifest is a declaration or allowlist entry, not evidence that it ran. A reader conclusion and a product validation result are product-side evidence, not platform-owned attestations. Missing, stale, conflicting, failed, or environment-blocked facts remain unresolved and must carry an owner or permitted follow-up.

Constraints

The sealed authority is read-only. It is revision `2026-08-08`, sets `employee_may_modify` to `false`, and lists exactly these logical source IDs: `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`. Each must resolve to a registry record with its locator, revision, digest, authority class, and `public-safe` disclosure. Only those public-safe references belong in this page's metadata; private governance references and source corpora do not.

The manifest is the route contract. Its command allowlist includes exact commands for parsing the authority, parsing the source registry, running the deterministic product-contract check, and parsing the lifecycle manifest. A declared command is not an observed result. An observed record must retain the exact invocation, exit status, stdout or stderr, time when available, source references, acceptance traces, and output location. If a result is not available, the absence is the evidence gap; prose must not fill it.

The lifecycle ends at a reversible candidate-only boundary. This page does not authorize approval, publication, promotion, rollback, activation, scheduling, release-pointer creation, or writing a live directory. It also does not edit the authority inputs, source registry, journey manifest, generated output, release directories, publication boundary, or platform-owned evidence.

Acceptance Checks

- **AC-1 — Lifecycle orientation:** The operator can state the Internal Knowledge context, audience, bounded objective, six stages, and product-evidence outcome.
- **AC-2 — Authority and disclosure trace:** The operator verifies the sealed authority revision, immutable employee flag, exact three logical IDs, and each registry record's public-safe metadata without copying private material.
- **AC-3 — Deterministic evidence distinction:** The operator uses an exact, non-empty allowlisted command and keeps its observed result separate from a declaration, reader evidence, and platform-owned attestation.
- **AC-4 — Reader evidence completeness:** The reader record names the journey ID, exact command, exit status, observed output, source references, acceptance traces, and output location; failed or unavailable evidence stays unresolved.
- **AC-5 — Independent review separation:** The packet identifies an independent reviewer and separate route, with route details, verdict, findings, and evidence location distinct from author conclusions.
- **AC-6 — Candidate-only handoff:** Only bounded evidence, acceptance traces, authority references, and unresolved findings reach the candidate boundary; no governed transition is inferred.
- **AC-7 — Reconciliation and repair boundary:** Every trace is reconciled, every unresolved fact has an owner or permitted follow-up, and missing or environment-specific evidence does not become a defect, repair target, claim change, or transition.

Lifecycle Stages

The contract groups the journey into six ordered stages. The responsible route and its evidence owner are explicit so a new operator can locate the next record without maintainer assistance. Stage 3 groups semantic authoring with the deterministic validation and projection path; it does not turn generated output or a command declaration into authority. Stages 4 and 5 deliberately separate reader conclusions from review conclusions and from any platform-owned execution or transition attestation.

Order	Lifecycle stage and responsible route	Required evidence artifact	Pass/fail gate	Next handoff
1	**Selection and bounded scope** — mission-selection and new-operator route	Workstream, audience, objective, entry point, and six-stage scope record, traced to the contract and manifest	**Pass:** the record names **Internal Knowledge**, the maintainer/operator/reader audience, bounded product scope, and candidate-only endpoint. **Fail:** keep selection unresolved if scope is broader or authority is implied.	Scoped packet to read-only authority tracing.
2	**Authority and public-safe source trace** — read-only authority route	`sources/authority.json`, `registries/sources.json`, and the source-reference comparison record	**Pass:** revision `2026-08-08`, `employee_may_modify: false`, the exact three logical IDs, and each public-safe registry mapping agree. **Fail:** a missing, stale, conflicting, or non-public-safe record blocks authoring.	Sealed reference metadata to product authoring.
3	**Semantic authoring, deterministic validation, and projection** — product author route followed by deterministic validator and projection inspector	This page, its registered page identity, the exact allowlisted command declaration, an observed command record when available, and projection inspection output	**Pass:** page metadata matches the registered identity, the command record preserves exact status and output, and projection identity is reproducible. **Fail/unresolved:** a declaration without an observed result, a mismatch, or missing projection record remains a product evidence gap.	Product evidence to independent reader evaluation.
4	**Fixed smoke replay and independent reader evaluation** — independent reader route	Fixed smoke replay and reader record keyed to the manifest journey with journey ID, exact command, exit status, observed output, source references, acceptance traces, and output location	**Pass:** the reader can trace AC-1 through AC-7 and keeps observations separate from author conclusions and platform attestations. **Fail/unresolved:** missing, failed, or environment-blocked smoke or reader evidence is carried forward with an owner or follow-up.	Reader record and unresolved findings to independent review.
5	**Independent review** — reviewer route separate from production	Review record naming reviewer, route, selected adapter/provider/model/role/policy when applicable, verdict, findings or an explicit empty set, and evidence location	**Pass:** the route is independent and the verdict is inspectable. **Fail:** no review pass is inferred from author prose, reader evidence, or a platform-owned record.	Review record and reconciliation matrix to handoff.
6	**Candidate-only handoff** — governed-cycle operator route	Bounded evidence packet, AC-1–AC-7 matrix, authority references, unresolved findings, owners or permitted follow-ups, and candidate-only disposition	**Pass:** all traces and gaps are reconciled and only the candidate boundary receives the packet. **Fail:** park it when a trace, owner, evidence boundary, or transition distinction is missing.	Reversible candidate-only packet for a separately governed decision.

For an LLM-backed production or review route, the run receipt must freeze the selected adapter, provider, model, role, and policy; production and independent review routes remain separate. Deterministic commands do not invoke an LLM. The page author may describe the evidence required by a route, but may not self-certify the route, its identity, its execution, or its transition result.

Required Artifacts and Gates

Use this matrix when assembling the packet. A required artifact can be a declaration, an observed result, a conclusion, or a platform-owned record; those categories must be labeled rather than merged. The exact command from the manifest is retained beside its result. For example, the deterministic product-contract declaration is `python3 -m pytest -p no:cacheprovider tests/test_product_contract.py -q`. That string alone proves only that the command is allowlisted. A pass requires an observed record with its exit status and output; otherwise the check is unresolved.

Checkpoint	Responsible route	Required artifact and ownership	Pass/fail interpretation	Next handoff
Scope selection	Mission-selection route	Workstream, audience, bounded objective, and entry-point record	**Pass:** the selected slice is Internal Knowledge and the endpoint is candidate-only. **Fail:** do not interpret later evidence for an unresolved scope.	Authority trace.
Authority trace	Read-only authority route	Authority JSON plus source-registry records for the three IDs; public-safe reference metadata only	**Pass:** revision, employee flag, IDs, locator, revision, digest, authority class, and disclosure agree. **Fail:** retain the source gap and stop authoring.	Semantic authoring.
Page identity	Product author route	This file and the existing `registries/pages.json` registration	**Pass:** page ID, title, public visibility, audiences, reading modes, empty claim references, and three source references match. **Fail:** reconcile the page source before any downstream check; this step does not edit the registry.	Deterministic validation.
Deterministic product check	Deterministic validator route	Allowlisted command declaration plus observed command record: exact invocation, exit status, counts or output, time if available, and location	**Pass:** an observed result supports the product conclusion. **Fail/unresolved:** a declaration, absent result, or failed result cannot be represented as a passing validator or platform gate.	Projection inspection and reader route.
Projection inspection	Deterministic projection route	Generated projection observation tied back to the semantic page and registry	**Pass:** identity and source references project reproducibly. **Fail/unresolved:** generated output is inspected as evidence and never edited as source.	Independent reader evaluation.
Reader journey	Independent reader route	Journey ID, exact allowlisted command, exit status, observed output, source references, AC traces, output location, and unresolved owner/follow-up when needed	**Pass:** all required fields are present and the reader conclusion is bounded to product reading. **Fail/unresolved:** missing, failed, or environment-blocked evidence remains explicit.	Independent review.
Independent review	Separate reviewer route	Reviewer identity, route and route details, verdict, findings or empty finding set, and evidence location	**Pass:** review is independent and inspectable. **Fail:** author conclusions cannot substitute for review, and neither can a platform-owned attestation.	Candidate reconciliation.
Candidate handoff	Governed-cycle operator route	Reconciled packet, AC-1–AC-7 matrix, authority references, unresolved findings with owners or permitted follow-ups, and candidate-only disposition	**Pass:** only bounded product evidence is parked. **Fail:** keep it parked when a boundary or trace is missing; do not infer approval or readiness.	Separate governed decision, if any.

The evidence labels are strict. A declared command is an allowlist fact; an observed command record is execution evidence. A reader conclusion says what the reader could establish from the page; a product validator conclusion says what the product-side check established. A platform-owned attestation is a separate fact supplied by its owning boundary. None may be substituted for another, and no absent result may be reconstructed from expected output.

Candidate-Only Handoff

The handoff packet contains only the bounded product material needed for a separate decision: the synchronized page identity, the three public-safe source references, authority and registry trace, deterministic command declaration and observed result or explicit gap, projection observation, reader record, independent review record, AC-1–AC-7 reconciliation, and every unresolved finding with its owner or permitted follow-up. The reader record must preserve `journey_id`, `exact_command`, `exit_status`, `observed_output`, `source_references`, `acceptance_traces`, and `output_location`. The review record must preserve reviewer, route, verdict, findings, and evidence location.

Candidate-only means parked, reversible, and incomplete as a governed transition. It is not approval, publication, promotion, rollback, activation, scheduling, release-pointer creation, or a write to a live directory. It is also not a readiness or platform-attestation claim. The operator transfers an unresolved gap exactly as observed, including its owner or permitted follow-up; the operator does not turn a missing reader result, failed review, or environment-specific observation into a defect, repair target, claim change, or pass.

The safe endpoint is singular: reconcile the bounded packet, place it at the candidate-only boundary, and stop. No command in this map authorizes approval, publication, promotion, rollback, activation, scheduling, release-pointer creation, or live-directory writes. Those actions, if ever considered, belong to separately governed authority and evidence boundaries outside this page.

Operator Trace

Follow this trace in order and keep each conclusion beside the evidence that supports it:

1. **Select and scope.** Record the Internal Knowledge workstream, intended audience, bounded objective, contract entry point, and candidate-only endpoint. Selection is context, not permission.
2. **Read authority.** Parse `sources/authority.json` and `registries/sources.json`; verify revision `2026-08-08`, `employee_may_modify: false`, the exact three logical IDs, and the public-safe locator, revision, digest, authority class, and disclosure for each.
3. **Author and check.** Compare this page's metadata with its registered page identity. Keep the exact deterministic command declaration separate from its observed record, inspect projection output without editing it, and leave any missing result unresolved.
4. **Replay and read independently.** Replay the fixed smoke journey using the journey manifest's exact command, then record the journey ID, command, status, output, source references, AC traces, and output location. State the reader conclusion only from that record; do not call it a platform attestation.
5. **Review independently.** Route the packet to an independent reviewer, preserve route details when applicable, and record verdict, findings, and evidence location. A reader conclusion and an author conclusion do not become review evidence merely by repetition.
6. **Reconcile and hand off.** Check AC-1 through AC-7, attach an owner or permitted follow-up to every unresolved fact, and transfer only the candidate-only packet. Stop before approval, publication, promotion, rollback, activation, scheduling, release-pointer creation, or live- directory writes.

The trace is complete when another operator can identify what was declared, what was actually observed, what the reader concluded, what the reviewer found, which facts remain platform-owned, and why the endpoint is candidate only. It is not complete when a command declaration is mistaken for a run, reader prose is mistaken for an attestation, or a missing fact is filled with an assumption.

Governed Evidence-Chain Failure Triage

Audience: agent-orch maintainers, workflow operators, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Governed Evidence-Chain Failure Triage

Purpose

This guide gives agent-orch maintainers and workflow operators a bounded way to triage an evidence-chain failure without guessing at its cause or owner. It separates a product repair from a validator defect, missing evidence, an environment dependency, and a Lee-authority decision. The reader should leave with the evidence to inspect, the owning route, the smallest immediate remediation, and a clear escalation boundary. The sealed authority is a source boundary only: revision `2026-08-08` lists the three public-safe logical source IDs and says `employee_may_modify: false`; it does not authorize changing governance, validators, routing, activation, approval, publication, or release state. Product conclusions and platform attestations must remain separate.

Classification Procedure

Begin with the original evidence record, preserving its journey or work-item identifier, exact command, exit status, observed output, source references, acceptance traces, retry or recurrence information, and affected work item. Do not replace a failed observation with a later pass. Next inspect `sources/authority.json` and match its exact source IDs to `registries/sources.json`; verify locator, revision, digest, authority class, and `public-safe` disclosure without copying private governance or runtime material. Then ask, in order: does a deterministic product result name a repairable input; does independent evidence show the validator itself is wrong; is a required evidence field absent; does the observation require an external environment; or is the remaining action a human authority decision?

Choose the narrowest class supported by observed evidence, not the most severe possible explanation. A missing field is missing evidence, not a product defect. `Operation not permitted` from the known localhost reader attempt is an environment-specific evaluation blocker, not proof of a recurring harness defect. A manifest command is a re-executable claim, not proof that a platform executed it. Record the class, inspected evidence, owner route, immediate remediation, unresolved fields, and escalation condition together so another reader can reproduce the classification.

The required synthetic journeys are intentionally independent of that localhost observation. They use allowlisted deterministic fixture checks over the guide and its journey manifest and expect exit code `0`; that result only shows that the local fixture contract was read and checked. It does not overwrite the preserved smoke attempt, whose observed command remains `python3 tests/check_documentation_smoke.py` with exit code `1`. The known environment-blocker and Lee-authority journeys retain that smoke command so their environment evidence stays distinct from the passing synthetic checks.

Failure Classes

The five classes below are mutually useful routing labels, not permissions to cross an authority boundary. Apply one only when the stated evidence exists; otherwise retain the observation as unresolved and request the missing record.

Product repair

Inspect the deterministic product result, the affected Markdown metadata, the matching page registry entry, registered source references, journey traces, and any projection mismatch. Use this class only when the evidence identifies a concrete product-input defect such as malformed structure, an unresolved reference, missing registration, or stale generated projection. The owning route is the product documentation or workflow maintainer. Immediately repair only the earliest failed product prerequisite, retain the original failure, and regenerate output from source when needed. Escalate when the source or authority boundary is inconsistent, no bounded product fix can address the observed result, or the product owner remains unresolved; never edit sealed authority to make a validator pass.

Validator defect

Inspect the exact validator command and version or policy context, its complete result, the product contract it claims to enforce, and an independent reproduction or known-good fixture. Use this class only when those records support an inconsistency in validator behavior, rather than a malformed input or a missing execution record. The owning route is the validator maintainer or platform boundary with authority over that validator. Immediately preserve the contradictory observations, stop treating the disputed result as a product conclusion, and request an independent validator review or corrected run. Escalate when validator authority, independent review, or a required gate capability is unavailable; do not rewrite content, claims, source records, or the authority seal to accommodate an unproven validator interpretation.

Missing evidence

Inspect the manifest, evidence packet, and source map for the journey ID, non-empty allowlisted command, integer exit status, observed output, source references, `traces_to`, retry or recurrence evidence, and affected work item. Use this class when the required record is absent, incomplete, stale, or conflicting and no supported failure cause can yet be established. The owning route is the evidence-producing journey, evaluator, or workflow that was required to capture the fact. Immediately mark the conclusion unresolved, preserve the partial record, and request a permitted capture with the exact allowlisted command; do not infer a pass, defect, retry count, or repair target. Escalate only when the missing owner cannot supply the evidence or a governed decision is blocked after the permitted request, carrying the missing fields and original observation into that escalation.

Environment dependency

Inspect the exact command, exit status, observed output, attempt count, reader service or network dependency, and platform-owned execution record. Use this class when the failure is caused by a required runtime capability outside the product source, such as the preserved `Operation not permitted` result from the evaluator environment denying localhost socket access. The owning route is the evaluator or platform environment owner. Immediately classify the observation as an environment blocker, retain it verbatim, and route a permitted follow-up evaluation where the required dependency is available. Escalate when the follow-up establishes a persistent capability gap or when the platform owner cannot provide the required attestation. One denied attempt does not establish a recurring product harness defect or authorize product mutation.

Lee-authority decision

Inspect the completed product conclusion, retained evidence chain, sealed authority metadata, `questions-for-the-human.md`, and any candidate-only boundary. Use this class when product repair and evidence routing are complete but the next action requires Lee's explicit advice or decision, such as the human-gate question about an unresolved environment-specific evaluation blocker. The owning route is Lee through the separately governed human gate. Immediately keep the candidate or work item parked, state the smallest decision requested, and preserve all evidence without inferring consent. Escalate to that human route only when the packet names the evidence and decision boundary; do not turn readiness, a validator pass, or silence into approval, pilot authorization, promotion, publication, scheduling, rollback, pointer mutation, or a live-directory write.

Remediation Routes

Route by the evidence boundary rather than by the filename that happens to contain the failure. For a product repair, the content or workflow maintainer owns the smallest source-side correction and the original observation remains beside the result. For a validator defect, the validator owner receives the contradictory command and independent reproduction; product authors make no compensating edit. For missing evidence, the journey or evaluator producer must capture the absent command result, source map, trace, recurrence, and affected work item. For an environment dependency, the platform evaluator must supply a permitted follow-up or an attestation of the capability gap. For a Lee-authority decision, the packet goes to the human gate with a narrow question and a reversible parked default. In every route, record `owner unresolved` when the supplied evidence names no owner, keep generated projections derived rather than hand-edited, and leave publication state unchanged.

Failure class	Evidence to inspect	Owning route	Immediate remediation	Escalate when
Product repair	Deterministic product result, page, registry, source refs, and projection	Product documentation or workflow maintainer	Fix the earliest failed source prerequisite and retain the failed result	The defect is outside the product surface or owner is unresolved
Validator defect	Exact validator result plus independent reproduction or known-good fixture	Validator maintainer or platform owner	Freeze the disputed conclusion and request independent review	Validator authority or independent review is unavailable
Missing evidence	Journey record, command, status, output, refs, traces, recurrence, and affected work	Evidence-producing journey or evaluator	Mark unresolved and request a permitted exact capture	Required evidence cannot be supplied or a gate remains blocked
Environment dependency	Runtime dependency, exact attempt, output, and platform execution record	Evaluator or platform environment owner	Preserve the blocker and route a permitted follow-up	Capability remains unavailable after governed follow-up
Lee-authority decision	Product conclusion, sealed authority, human-question record, and candidate state	Lee through the human gate	Park the work and ask the smallest explicit question	The named human decision is required to resolve the boundary

Escalation Boundaries

Escalation is required only after the record is complete enough to show what was observed, what was checked, and why the current owner cannot resolve the gap within its boundary. Escalate a product issue to its product owner when a concrete deterministic defect is outside the current bounded repair. Escalate to validator or platform ownership when independent evidence identifies a validator inconsistency or an environment capability gap. Escalate missing evidence to the producer when a required field, command result, recurrence, source reference, or affected work item prevents a supported conclusion. Escalate to Lee only when the packet presents a completed evidence chain and a smallest human decision, not merely because a check was not run.

The known reader-gate observation remains narrow: the one `journey.resolve-reader-journey-harness-decision` attempt used `python3 tests/check_documentation_smoke.py`, exited `1`, and reported `Operation not permitted` because localhost socket access was denied. That is an environment-specific evaluation blocker. It does not establish recurrence, a product harness defect, an affected claim, or a repair target. Preserve the candidate-only and unresolved default until a permitted follow-up supplies those facts. No escalation may modify sealed authority, private governance, validator authority, routing, activation, approval, publication, promotion, scheduling, rollback, a release pointer, or a live directory.

Acceptance Checks

- **AC-1 — Correct classification:** Given a known or synthetic evidence-chain record, the reader identifies the narrowest supported class among product repair, validator defect, missing evidence, environment dependency, and Lee-authority decision; cites the exact evidence inspected; and keeps an unsupported cause, recurrence, defect, or repair target unresolved.
- **AC-2 — Correct owner or remediation route:** The reader names the owning product, validator, evidence-producing, environment, or Lee route and states the smallest immediate remediation, including preservation of the original observation and the required exact command, status, output, references, and traces. An unprovided owner is recorded as `owner unresolved` rather than invented from a filename or generated artifact.
- **AC-3 — Correct escalation boundary:** The reader states when escalation is required, carries the complete evidence chain into that route, and preserves the reversible default when proof is incomplete. The reader does not infer a platform attestation, human approval, promotion, publication, scheduling, rollback, pointer change, or live-directory write from a product check, missing evidence, or one environment-denied attempt.

Governed Execution Engine

Audience: owner, onboarding-teammate, prospective-client | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Governed Execution Engine

The Governed Execution Engine documentation is maintained as one semantic source and evidence model. Reading modes may change emphasis and presentation, but they do not create competing versions of a platform claim.

This overview keeps the documented claim aligned with its source references and evidence model. It describes the documentation structure for public-safe reading modes, so the same claim can be read with executive, engineering, or AI emphasis without adding a separate platform claim or unsupported outcome.

How to read this page

Read the page as one traceable explanation. Executive mode highlights the governing idea; engineer mode makes the source and evidence relationships explicit; and AI mode exposes that same structure for consistent interpretation. These are reading modes over one semantic source and evidence model, not separate claims or competing documents.

Start here

If you have no repository context, begin here. First locate the governing source boundary in `sources/authority.json`, then open `registries/sources.json` and match `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract` to their locators, revisions, digests, and `public-safe` disclosures. Next read the `Semantic source`, `Projection`, and `Release boundary` sections of `architecture.md` to understand how `content/`, `registries/`, and `models/` relate to authored Markdown and the deterministic `build/index.json` projection. The `source.platform-delivery-contract` record is the contract governing this delivery profile's reproducibility; the reproduced evidence is `build/index.json` together with the deterministic content-validation and projection checks. Finally read the `Employee Onboarding: First Task contract` and reader manifest for this route before completing the named first documentation task with: `python3 tools/validate_content.py --journey journey.employee-onboarding-first-task --semantic-only`. This produces product-side documentation evidence only; it grants no platform authority and performs no approval, publication, promotion, rollback, release-pointer, or live-directory action. Keep missing or conflicting source facts unresolved.

For prospective adopters

Prospective and current customers can use this overview to understand the public-safe value of a governed documentation surface: one semantic source and evidence model is presented consistently across executive, engineering, and AI reading modes. The modes change emphasis, not the underlying claim. This page helps a reader assess whether a bounded documentation use case is worth evaluating; it does not promise a capability, business outcome, performance, compliance result, availability, or customer result. Keep any missing, stale, or conflicting authority visible as an unresolved gap.

Trace the governing claims

Start with `sources/authority.json`, whose approved source list is `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`. In `registries/sources.json`, verify each ID, its recorded locator and revision, and its `public-safe` disclosure: `source.naming-decision` maps to `factory:decisions.md#Governed Execution Documentation Steward naming and route`; `source.documentation-boundary` maps to `product:architecture.md#Semantic source`; and `source.platform-delivery-contract` maps to `agent-orch:docs/delivery-profile-contract.md#content`. The one-semantic-source and evidence-model claim and its reading-mode boundary trace to all three IDs; the no-publication boundary traces to the documentation-boundary and platform-delivery-contract IDs. If an ID, revision, digest, or disclosure is missing or stale, record the claim as unresolved rather than guessing.

Plan your evaluation

Select one bounded documentation use case, write down the claims and evidence it requires, and compare that list with this overview and the three approved registry records. Then request or conduct one scoped evaluation conversation about that comparison. Keep the next step reversible and product-scoped: the conversation evaluates fit and evidence needs, not approval, publication, release-pointer creation, a promised capability, or an adoption result. Record the specific unresolved gaps and stop there if the approved evidence does not establish a conclusion.

Publication Boundaries and Reader Journeys

A reviewed candidate is evidence that a bounded package is ready for review, not evidence of approval or publication. Candidate status remains distinct from promotion, scheduling, rollback, a release pointer, and a live directory. To trace a claim, start at `sources/authority.json`, follow its public-safe IDs through `registries/sources.json`, and use `build/index.json` with the deterministic content-validation, projection, and smoke evidence to reproduce the documented source and evidence trail. For version interpretation, compare the immutable candidate record's previous-version identity, rollback-target identity, rollback availability, and any release-blocking fact; if a fact is missing, stale, or conflicting, record it as unresolved rather than inventing one. Keep product conclusions separate from platform-owned attestations. The next adoption decision is therefore reversible and product-scoped: select one bounded use case, review its evidence gaps, and keep the candidate parked while fit is evaluated. This guidance creates no pointer, promotes nothing, writes no live directory, and does not publish, schedule, or roll back anything.

Candidate-Evidence Preflight Troubleshooting

This is a bounded **Internal Knowledge** troubleshooting route for auto-orch operators and maintainers. Before interpreting any candidate-evidence failure, open `sources/authority.json` and verify `authority_revision: 2026-08-08`, the exact IDs `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and `employee_may_modify: false`. Match each ID in `registries/sources.json` and check that its record is disclosed as `public-safe`, with its recorded locator, revision, and digest. Those records map respectively to `factory:decisions.md#Governed Execution Documentation Steward naming and route`, `product:architecture.md#Semantic source`, and `agent-orch:docs/delivery-profile-contract.md#content`. A missing, stale, or conflicting authority or registry fact stays unresolved; it is not repaired by editing prose, fixtures, manifests, or protected sources.

Then use the selected `journeys/candidate_evidence_preflight_troubleshooting.json` as the route declaration for this page. It is mission-authorized, candidate-only, tied to this page, and declares five required, non-exploratory journeys with acceptance traces and exact commands. The contract, model, preflight tool, tests, and fixtures explain the product-side preflight shape; they do not add platform authority. A declaration, expected result, or named review path is not evidence that a command ran or that independent review exists.

Failure Modes

Classify the first observed error before remediation. A schema or evidence-shape failure concerns the required top-level fields `schema_version`, `kind`, `journeys`, and `review_verdict_path`, or a journey's `journey_id`, `name`, `status`, `steps_taken`, `source_refs`, `traces_to`, `blockers`, and `commands_run`. Each command claim also needs `command`, integer `exit_code`, and non-empty `observed_result`; coverage, trace mapping, and a missing observed field are separate failures. A command-allowlist failure means the claimed string is not an exact member of the selected manifest's allowlist. A journey-identity failure means the ID, name, position, or trace does not match the selected declaration. Review-path failures concern the exact `code-reviews/content-review.verdict.json` path and its independently readable artifact. A fixture illustrates a validator case; it is not execution evidence.

For this route, compare commands byte-for-byte against this exact allowlist:

```
python3 -B -m json.tool sources/authority.json
python3 -B -m pytest -p no:cacheprovider tests/test_candidate_evidence_preflight.py
python3 -B -m json.tool journeys/candidate_evidence_preflight_troubleshooting.json
python3 -B tools/validate_content.py --semantic-only
```

The selected journey command must also equal that journey's own declaration. Leading or trailing spaces, doubled spaces, newlines, alternate executables, or a command that merely succeeds elsewhere do not match. An expected result does not replace the actual observed result. Keep the candidate-only boundary and the separation between product conclusions and platform-owned attestations; neither a failure category nor a successful check is transition authority.

Compare journey names byte-for-byte as well. The five selected names are:

```
As an auto-orch operator, verify the sealed authority and public-safe source boundary before troubleshooting.
As an agent-orch maintainer, diagnose candidate-evidence schema and observed-field failures.
As an auto-orch operator, distinguish exact command-allowlist and journey-name failures.
As a future playbook author, separate fixture examples from independent review-path evidence.
As an independent reader, verify the troubleshooting guide and route unresolved evidence without crossing the candida
```

Do not shorten, punctuate, normalize, or replace a name with an ID. A name or command mismatch remains attached to the original journey and is escalated as declaration drift.

Diagnosis and Remediation

Read `docs/candidate-evidence-preflight-contract.md`, `models/candidate_evidence_handoff_contract.json`, and `tools/candidate_evidence_preflight.py` together with the relevant test and fixture. For schema failures, compare the record with the model and tool: top-level `schema_version: 1`, kind `candidate-evidence-preflight-evidence`, the complete declared journey set, and `review_verdict_path`; then verify each journey's exact name, status, non-empty `steps_taken`, `source_refs`, `traces_to`, and `blockers`. The command claim must preserve the exact declaration, an integer `exit_code`, and a non-empty `observed_result`. The schema permits `passed`, `failed`, `blocked`, and `unresolved`, while the preflight rejects a record whose journey status is not `passed`. Preserve the actual output, status, and blocker; never fill a missing observation with an expected result.

For command drift, compare the complete claimed string byte-for-byte with both the journey declaration and the four-entry allowlist above. The test definitions and `disallowed-command.json` specify a non-allowlisted command as a rejection case, even when it is another plausible content command; that declaration is not proof that a test ran. The same definitions specify leading or trailing spaces, doubled spaces, and newlines as non-matches; there is no normalization step. If a needed check is outside the allowlist, preserve that mismatch and route a manifest or contract decision to its owner instead of broadening the page or candidate record.

For journey drift, compare `journey_id`, exact `name`, `traces_to`, and array position with the selected manifest. `invalid-journey-name.json` encodes the case where a correct ID cannot rescue a shortened name; it does not record an executed failure. The selected troubleshooting manifest declares five required journeys and its four-command allowlist. The human-readable contract carries a separate set of five canonical names and a two-command allowlist, while the model, executable preflight, and supplied handoff fixtures name `journeys/user_journeys_manifest.json` and the three-journey handoff declarations. These are not interchangeable routes. Recreate evidence only from the current selected declaration and keep any mismatch, path, field, value, expected code, and owner unresolved; do not choose whichever declaration produces a pass.

For the supplied fixture cases, first identify the declaration set they carry. `valid.json` is a complete-shape example for the three-journey handoff shape; it is not automatically valid evidence for this selected five-journey route. `malformed-evidence.json` omits `observed_result`, `disallowed-command.json` uses `python3 tools/validate_content.py`, `invalid-journey-name.json` changes the second exact name, and `missing-review-path.json` omits `review_verdict_path`. These files encode validator cases only; none is execution evidence. The required review artifact is exactly `code-reviews/content-review.verdict.json`; independently verify that it is readable JSON with `schema_version: 1`, a `verdict` of `pass` or `clean`, a `findings` array, and no `High` or `Critical` finding. A named path, fixture shape, or local declaration does not fabricate independent review; preserve a missing, stale, malformed, or unverified verdict as a blocker.

Verification and Escalation

After authority verification, an authorized operator may use only the selected journey's exact allowlisted command. The reproducible record must preserve the repository path, journey ID and byte-exact name, command string, integer exit code, observed output, `steps_taken`, `source_refs`, `traces_to`, `blockers`, and the owner of any unresolved field. A command declaration is not an assertion that it ran, and an expected result is not an observed result. The

fixtures are deterministic rejection examples, not execution evidence; the valid fixture is a shape example, not a review result. The exact independently readable review path and its verdict fields must be checked separately.

Escalate when authority or registry data, the selected manifest, contract, model, executable preflight, exact journey name, allowlisted command, observed field, fixture interpretation, or review artifact cannot be reconciled. Include the path, exact field or command, observed value, expected declaration, preserved output, and owner—or explicitly record ownership as unresolved. A failed observation alone does not establish a recurring defect, retry count, affected claim, repair target, route, validator attestation, or gate result. Product-side conclusions remain separate from platform-owned attestations. Keep the handoff parked and candidate-only while the governing owner resolves the declaration or supplies independent evidence. Nothing in this troubleshooting result authorizes approval, activation, promotion, publication, scheduling, rollback, a release pointer, or a live-directory write.

Acceptance Checks

- **AC-1:** Verify authority revision `2026-08-08`, the three exact authority source IDs, and each matching `public-safe` registry record before interpretation; do not infer or modify authority.
- **AC-2:** Diagnose top-level, journey, command-claim, observed-field, status, and coverage failures from the contract, model, tool, tests, and fixtures; preserve actual values and unresolved blockers.
- **AC-3:** Compare each command and each journey name byte-for-byte with the selected manifest and its allowlist; retain the original identity and route declaration drift as unresolved.
- **AC-4:** Distinguish fixture shape examples from execution evidence and independently verify the exact review path and accepted verdict shape; missing, malformed, stale, or unverified review remains a blocker.
- **AC-5:** Record reproducible checks with paths, commands, exit codes, observed output, traces, and ownership, then escalate within the candidate- only boundary while keeping product conclusions separate from platform attestations and every approval or publication transition.

Governed Orchestration Buyer Decision

Audience: prospective-buyer | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Governed Orchestration Buyer Decision

Buyer Decision

The concrete buyer question is whether governed orchestration provides a trustworthy, appropriately bounded way to run unattended work. The approved source boundary supports a narrower decision: continue a reversible evaluation of whether the documentation's provenance and evidence route can be traced without an unresolved disclosure concern. It does not establish that unattended work can run, that execution will be controlled or enforced, or that the product is ready for adoption. A buyer may therefore treat traceability as an evaluation criterion, but must keep the runtime decision open until separate authorized evidence addresses it. “Trustworthy” here means that the boundary and the evidence status are stated plainly; it is not a security, reliability, compliance, performance, availability, or customer-results guarantee. The appropriate outcome is a scoped go-forward for evidence review, not a purchase, deployment, approval, publication, promotion, scheduling, rollback, or release decision.

Decision in plain terms: yes, the approved evidence supports continuing a bounded unattended-work evaluation when “evaluation” means reviewing documentation provenance and evidence traceability. No, it does not support running unattended work, claiming that execution is governed, or adopting the product. That distinction is the stopping rule for this page.

What Governed Orchestration Changes

For this review, governed orchestration changes the decision frame rather than proving a new runtime capability. It makes the buyer ask what the sealed source boundary can support, what an evaluator can observe, and where the evidence must stop. The authority names exactly three logical source IDs and the source registry supplies the permitted public-safe reference fields for those IDs. That makes provenance and disclosure alignment reviewable. It does not turn source metadata into an execution attestation. The same distinction applies to the reader check: the evaluator can inspect the documentation surface and record the command's actual result, while the result remains documentation evidence. It cannot be used here to infer unattended execution, policy enforcement, isolation, security, operational availability, pricing, customer benefit, or deployment status. The change for a buyer is a better-bounded question and a clear stopping rule: proceed with traceability evaluation only when the approved records align, and request new authorized evidence before making a broader runtime or adoption decision.

Evidence and Limits

The sealed authority is `sources/authority.json` at revision `2026-08-08`; it lists `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and records `employee_may_modify` as `false`. The corresponding registry entries are the approved public-safe reference boundary. Their disclosed fields are source ID, locator, revision, digest, authority class, and disclosure value:

Source ID	Authority class	Revision	Disclosure
`source.naming-decision`	`canonical-platform-source`	`2026-08-02`	`public-safe`
`source.documentation-boundary`	`product-owned-public-safe-source`	`2026-08-04`	`public-safe`
`source.platform-delivery-contract`	`canonical-platform-source`	`agent-orch-main-e65a64fcf429`	`public-safe`

These records authorize disclosure of reference metadata, not disclosure of a private source corpus or runtime evidence. This page has no claim references, because the sealed inputs do not establish a capability claim. Nothing here reports deployment, publication, pricing, a customer outcome, a security guarantee, performance, availability, compliance, or successful unattended execution. The independent evaluator should run `python3 tools/validate_content.py` and record its actual exit status and observable output. This deterministic repository check is documentation evidence: it verifies the public page and

registry/source contracts and the reproducible projection, but it is not a runtime attestation. A missing, stale, conflicting, or non-public-safe record ends the supported traceability conclusion rather than being repaired with an inferred source or a stronger product statement.

Decision Path

1. Inspect `sources/authority.json`. Confirm revision `2026-08-08`, the exact three authority-listed IDs, and `employee_may_modify: false`.
2. Resolve every ID in `registries/sources.json`. Confirm that each record is present, its disclosed reference fields are readable, and its disclosure is `public-safe`. Do not copy or reconstruct a source corpus.
3. Compare this page with its entry in `registries/pages.json`: the page ID, title, public visibility, prospective-buyer audience, reading modes, empty claim references, and three source references must agree.
4. Run `python3 tools/validate_content.py` and record the actual exit status and observable output. Treat the deterministic result as documentation evidence, not a runtime attestation.
5. If the authority, source disclosures, page metadata, and evaluator record are aligned, answer yes to continuing the bounded unattended-work evaluation described above, limited to documentation provenance and evidence traceability. If any part is unresolved, record the concern and stop the traceability conclusion.

In either case, leave the decision to run unattended work open. A buyer still needs separately authorized evidence for execution behavior, control or enforcement, operational performance, security, availability, compliance, deployment, pricing, customer outcomes, and any adoption commitment. This path does not authorize a transition or imply that one has occurred.

Acceptance Checks

- **AC-1 — Authority and synchronization:** The reader verifies authority revision `2026-08-08`, exactly the three listed logical source IDs, `employee_may_modify: false`, and a matching `public-safe` record for each. The reader also confirms that this page and its registry entry agree on ID, title, visibility, audiences, reading modes, claim refs, and source refs.
- **AC-2 — Bounded evaluator evidence:** The reader runs `python3 tools/validate_content.py`, records its actual exit status and observable output, and identifies the deterministic result as documentation evidence only. The reader does not infer unattended execution, enforcement, security, customer outcome, deployment, approval, or publication from it.
- **AC-3 — Buyer decision boundary:** When the approved source records are present, public-safe, and synchronized with the page, the reader answers yes to continuing a bounded unattended-work evaluation only when that evaluation is limited to documentation provenance and evidence traceability. The reader answers no to treating the records as support for running unattended work or broader adoption, keeps the unattended-runtime decision unresolved, and takes no deployment, publication, pricing, or other transition action.

Journey-Manifest Failure Gap Matrix and Checklist

Audience: Lee, future playbook authors | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Journey-Manifest Failure Gap Matrix and Checklist

Purpose and Audience

The selected workstream is **Internal Knowledge**. This page is for **Lee and future playbook authors** who need a compact way to find an uncovered journey-manifest remedy and identify the operational check that can show whether the remedy is addressed. Its intended outcome is quicker identification of uncovered journey-manifest remedies and required operational checks, while keeping the source boundary explicit and leaving unsupported facts unresolved. The page uses only the three logical sources sealed by `sources/authority.json` and recorded as `public-safe` in `registries/sources.json`.

The sealed authority revision is `2026-08-08`. It names `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and sets `employee_may_modify` to `false`. Those records are inputs, not prose to reinterpret. A gap in this matrix means that a remedy, its verification, its owner, or its decision coverage is not explicit in the bounded product artifacts. It does not establish platform behavior, a historical outcome, or any decision that is absent from the sealed inputs.

Failure-Gap Matrix

Use one row per uncovered condition. The owner column names an owner only when the bounded inputs support one; otherwise it records that the decision is missing. A command is a reproducible check to be run by an authorized evaluator, not evidence that it has already run. Evidence references remain limited to the sealed authority and source registry plus the three product artifacts in this step.

Identified failure gap	Practical remedy	Verification command or evidence	Owner	Covered decision or explicit missing decision
The page or journey manifest could drift from the sealed source set, or name a source that is not one of the three authority-listed records.	Compare the page metadata and journey source references with `sources/authority.json` and `registries/sources.json`; remove any reference that is not one of the three `public-safe` records.	Evidence: the authority `source_files` array and matching registry records. Command: `python3 -B tools/validate_content.py --semantic-only`.	Not established in the sealed inputs; do not infer an owner for changing the authority boundary.	Covered by AC-1. The sealed inputs do not record a decision authorizing a fourth source, so no fourth source may be inferred.
The page header and its `registries/pages.json` entry could disagree on identity, title, visibility, audience, reading modes, claim references, or source references.	Copy the page header values exactly into the single registry entry and keep the claim list empty; then inspect the pair as one synchronization unit.	Evidence: the page metadata block beside the matching registry object. Command: `python3 -B tools/validate_content.py --semantic-only` when an authorized evaluator is permitted to run the deterministic product check.	Not established in the sealed inputs; do not infer an owner for the bounded synchronization decision.	Covered by AC-1. The matrix records no authority to change the source registry or authority record when synchronization exposes a source decision gap.
A journey can declare a command without showing that the command belongs to the reproducible allowlist or that its trace is covered by an acceptance check.	Keep the allowlisted command in the journey manifest, use only that command in non-exploratory journeys, and make `acceptance_checks` and `traces_to` resolve to AC-1, AC-2, or AC-3.	Evidence: the journey manifest's `command_allowlist`, each journey command, and matching acceptance traces. Command: `python3 -B tools/validate_content.py --semantic-only`.	Not established in the sealed inputs; the owner of recording an executed result is an explicit missing decision.	Covered by AC-2. The missing decision is who records an executed result when a command declaration is valid but no reader evidence is supplied.
Deterministic product checks, reader evidence, and independent review could be conflated, causing an unexecuted command or a reader statement to be treated as proof.	Label each check by evidence class, retain the exact command and observed result for reader evidence, and have an independent reviewer compare the page, registry, and journey traces without adding facts.	Evidence: a deterministic check result, a reader record with command and output, and an independent comparison of the three bounded artifacts. Command: `python3 -B tools/validate_content.py --semantic-only` for the deterministic product check.	Not established in the sealed inputs; the evidence owner and reviewer identity remain explicit missing decisions.	Covered by AC-2. No execution result, reviewer identity, or acceptance threshold is invented by this page.
A failed or uncovered check could lack an explicit next decision, remedy boundary, or owner, leaving authors tempted to fill the gap with unsupported platform or historical claims.	Preserve the gap as unresolved, state the smallest evidence needed to close it, and route a bounded decision that names the owner, allowed evidence, and affected artifact before changing content.	Evidence: the unresolved row itself plus the later decision record, if one is supplied. Do not substitute a guessed result for missing evidence.	Not established in the sealed inputs; Lee is the stated audience for this page, not an inferred owner of a new remedy decision.	Covered by AC-3. Explicit missing decisions remain missing; this page does not convert them into approvals, platform facts, or historical outcomes.

The matrix is complete only when every identified gap has a practical remedy, an executable command or named evidence, an owner or an explicit unresolved owner, and either a covered decision or a clearly recorded missing decision. Adding detail that cannot be tied to the three sealed public-safe sources is a new

claim and is outside this page's bounded repair route.

Operational Checklist

Classify the check before recording its result. Deterministic product checks inspect the page, registry, and journey structures and may use only the reproducible commands declared in the journey manifest. Their result speaks to the product artifacts, not to a platform execution. Reader evidence records the exact journey, command, exit status, observed output, source references, and acceptance trace supplied by the reader; a command declaration alone is not reader evidence. Independent review is a separate comparison by a reader who checks the matrix rows, metadata synchronization, source boundary, and AC-1 through AC-3 traces without silently resolving a missing decision.

Use this sequence for each gap:

1. **Deterministic product checks:** where an authorized evaluator is permitted to run it, use the exact allowlisted command `python3 -B tools/validate_content.py --semantic-only` for the bounded product contract check. Record the command and result separately from interpretation; the command declaration itself is not an execution result.
2. **Reader evidence:** record the journey ID, exact allowlisted command, exit status, observed output, source or artifact references, and the AC-1, AC-2, or AC-3 trace. If a field is unavailable, mark it unresolved rather than replacing it with an inferred result.
3. **Independent review:** compare the page header with `registries/pages.json`, compare every journey command with its allowlist, confirm that the three authority-listed sources are the only source references, and verify that every non-exploratory journey traces only to AC-1, AC-2, or AC-3.

For the uncovered remedy selected from a matrix row, verify these evidence fields together: the identified failure gap, practical remedy, exact allowlisted verification command or named evidence, owner or explicitly unresolved owner, and acceptance trace. A reader may verify that each field is present in the row and its journey trace; the execution status and observed output remain unresolved until a separate reader record supplies them.

The checklist is operationally complete when these three evidence classes are visibly distinct and every gap row points to the evidence class that can verify its remedy. A review may identify a missing decision, but it must not silently supply one.

Missing Decision Escalation

Escalate only the unresolved decision that prevents a row from being closed. The sealed inputs do not supply a new source, a remedy owner, an independent reviewer, an execution result, or an acceptance threshold beyond the bounded records described here. Therefore, an author must not choose one of those facts by reading intent into a locator, a command string, or an empty claim list. Mark the field as a missing decision and preserve the exact gap.

A useful escalation record names the affected matrix row, the three sealed source references consulted, the exact allowlisted command or evidence needed, the artifact to be checked, the proposed decision owner, and the acceptance criterion that would close the row. Until that record exists, keep the remedy unassigned or unresolved and make no content change that would imply a source addition, a platform result, a historical result, or a decision that the sealed inputs do not contain. The smallest safe follow-up is to obtain the missing bounded decision or evidence, then update only the affected row and its synchronized page or journey artifact.

If the missing field is ownership, ask who is authorized to record the deterministic result, reader evidence, or independent review. If it is evidence, ask which exact command or artifact is acceptable. If it is decision coverage, ask which of AC-1, AC-2, or AC-3 closes the row. These questions keep the escalation factual and prevent an unresolved product gap from being recast as platform behavior or a historical outcome.

Acceptance Checks

- **AC-1 — Matrix coverage:** Every identified failure gap has a practical remedy, a verification command or named evidence, an owner or explicitly unresolved owner, and either a covered decision or an explicit missing decision. The row references remain within the three sealed public-safe source records and the bounded product artifacts.
- **AC-2 — Executable operational checks:** The checklist distinguishes deterministic product checks, reader evidence, and independent review; each selected remedy names its failure gap, practical remedy, exact allowlisted verification command or named evidence, owner status, and acceptance trace; every non-exploratory journey uses a command from the reproducible allowlist; and no declared command is presented as execution evidence without a recorded result.

- **AC-3 — Explicit missing-decision handling:** Any absent owner, evidence, reviewer, threshold, or remedy decision remains marked unresolved and is routed with the exact gap, source references, artifact, command or evidence, and proposed decision needed for closure. No platform behavior, approval, publication fact, or historical outcome is inferred.

New Contributor Evidence-Chain Walkthrough

Audience: new agent-orch contributors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

New Contributor Evidence-Chain Walkthrough

Overview

This walkthrough gives a new agent-orch contributor one representative task: trace the source authority for a bounded onboarding page, compare its authored metadata with the synchronized page registry, and read the accompanying journey manifest. The task is about following evidence roles in order, not about asserting a platform outcome. The page is public-safe because it uses only the three approved logical sources: `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`.

The starting boundary is the read-only `sources/authority.json` record. It identifies authority revision `2026-08-08`, lists exactly those three logical source IDs, and records `employee_may_modify: false`. Their matching entries in `registries/sources.json` are the public-safe source boundary. If a source fact is absent or conflicting, the contributor keeps it unresolved rather than filling the gap with plausible prose or private governance material.

Trace the Sealed Evidence Chain

First, open `sources/authority.json` and follow each of its three IDs to the corresponding public-safe record in `registries/sources.json`. This is source provenance for the walkthrough, not permission to change authority or any separate governing control. Check that the page metadata and its one registry entry carry the same page ID, title, visibility, audiences, reading modes, empty claim references, and exactly the same three source references.

Next, distinguish the preserved deterministic validation from `build/index.json`. The validator record remains authoritative for its exact command and result; this walkthrough does not replace it with a worker assertion. `build/index.json` is generated projection output from the authored product sources; it is useful for inspecting the projection, but it is not semantic source, validator authority, approval, or a publication pointer. A later projection result must not replace preserved validation evidence or turn an unobserved result into a claim.

Then, distinguish reader evidence from independent review. Reader evidence records what a reader observed while following a named journey and has its own owner and trace. Independent review is a separate record, with its own reviewer, route, owner, and trace, that assesses the product conclusion. Neither record self-certifies the other, and neither reader or reviewer conclusion becomes a platform-owned execution or transition attestation. Keep those ownership boundaries visible when the task is handed off.

Finally, carry the evidence into the candidate-only boundary. A candidate is a bounded product package that can be examined while its facts remain unresolved; it is not evidence that a transition occurred. The contributor stops after recording the product evidence and does not infer a missing release fact from the page, registry, projection, reader result, or review result.

Candidate-Only Boundary

Candidate-only means parked for review within the product evidence boundary. A candidate is not publication, promotion, creation or movement of a release pointer, or rollback. It also does not imply approval, scheduling, activation, or a write to a live directory. Those actions have separate owners and are not performed by this walkthrough. The reader must keep prior-version identity, rollback-target identity, and rollback availability unavailable whenever no authoritative record establishes them; a null or missing fact is not a license to invent one.

The sealed source authority, deterministic validation, generated projection, reader evidence, and independent review each answer a different question. They can support a bounded product conclusion about this documentation route, but they do not enlarge the source list or create transition authority. The safe finish is therefore a synchronized page and reader contract with every gap visible, while publication, promotion, pointer, and rollback remain outside the task.

Acceptance Checks

- **AC-1:** Verify authority revision `2026-08-08`, the exact three logical source IDs, and `employee_may_modify: false`; match all three IDs to public-safe registry records. Treat this as read-only source provenance and do not copy private governance material or add a source.
- **AC-2:** Distinguish the deterministic validation result from generated `build/index.json`. Keep the projection separate from semantic source, validator authority, approval, and publication-pointer status; do not use generated output to replace preserved validation evidence.
- **AC-3:** Identify reader evidence and independent review as separate records with separate ownership. Report product conclusions only within this walkthrough and keep platform-owned execution and transition attestations outside reader or reviewer self-certification.
- **AC-4:** State that a candidate-only package is not publication, promotion, release-pointer creation, or rollback. Preserve unavailable prior-version and rollback facts as unavailable, and confirm that this task exercises no approval, scheduling, activation, live-directory, or other transition action.

No Additional Work Cycle

Audience: owner, onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

No Additional Work Cycle

Decision

No additional documentation work was selected for this cycle because the sealed material available to this workspace did not identify an evidence-backed craft gap or a target module that could be responsibly repaired. This is a narrow product conclusion about the available evidence, not a new authority decision and not an instruction to alter the separate governing repository, its routing, validation, activation, or publication controls.

Evidence

The inspected source boundary is `sources/authority.json`, revision `2026-08-04`, listing the logical source files `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`. `sources/approved/README.md` explains that logical references and digests live in the registries and that private source corpora and runtime evidence are absent here. The available material therefore supports a bounded no-work conclusion, but does not support inventing a gap, target, claim, authority, or broader scope.

Reader Outcome

An operator can verify what was decided, why the available evidence was insufficient to select documentation work, and what remains reversible. The next action is to retain this record and, only if newly approved source material later names a concrete gap or target module, open a fresh bounded cycle and reassess. The reader should not convert an evidence absence into speculative content, and should understand that this record is not a release candidate or a publication instruction.

Acceptance Checks

- AC-1 — The record states that no evidence-backed craft gap or target module was available in the sealed material, so no documentation target was selected.
- AC-2 — The record does not invent claims, source contents, authority, or scope; it identifies the available paths and revision without granting powers those sources do not grant.
- AC-3 — The reader can identify the reversible next action: retain the record and reassess in a fresh bounded cycle if approved evidence later names a gap or target module.
- AC-4 — Publication, promotion, rollback, and live-directory writes did not occur as part of this no-work decision; this record is not a release candidate or publication pointer.

Operator Decision-Gap Analysis

Audience: Auto-Orch operators, reviewers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Operator Decision-Gap Analysis

Purpose and scope

This public-safe page is a bounded Internal Knowledge aid for Auto-Orch operators and reviewers. It explains which narrow decision a candidate-handoff preflight observation can support, which evidence must be inspected, and which single decision aid remains missing. It is read-only guidance: a declaration, expected result, or absent fact is not an execution attestation, product defect, approval, repair target, or transition authority. The sealed authority revision is `2026-08-08`; its three public-safe source IDs are `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, with `employee_may_modify: false`.

Preflight observation

The current handoff manifest declares the documentation-validation route as `python3 -B tests/check_documentation_smoke.py` with expected exit code `1`. That declaration is not evidence that the command ran. The supplied reader-gate observation instead records one attempt of `journey.resolve-reader-journey-harness-decision` using `python3 tests/check_documentation_smoke.py`, exit status `1`, and observed `Operation not permitted` because the evaluator environment denied localhost socket access. The no-`-B` observation must not be silently normalized into the allowlisted command. It describes an environment-specific evaluation blocker; it does not establish a recurring product harness defect, retry count, affected claim, or repair target.

Supported decision

The evidence supports one bounded documentation decision: keep the candidate-only handoff unresolved and identify a permitted follow-up evaluation as the missing evidence route, while preserving the original observation. A later record must retain the journey ID, exact allowlisted command, exit status, observed output, steps, source references, and acceptance traces. This page does not run that evaluation, select a repair, approve a handoff, or authorize any workflow transition. A product-contract check can support internal consistency only; it cannot substitute for a platform-owned execution or gate attestation.

Uncovered decision aid

One decision aid remains uncovered: an authoritative recurrence-and-impact record that distinguishes a product repair route from an environment-specific evaluation blocker. It must supply the journey ID and exact command, exit status and observed output, retry count or other recurrence evidence, relevant source references, affected work or claim, and the owning product, evaluator, or platform route. Those facts are unresolved. Do not infer a defect, affected claim, owner, or repair recommendation from the single localhost-denied attempt, from an expected exit code, or from a result produced by another route. The synchronized reader-journey manifest records the three required reader journeys and their acceptance traces.

Read-only boundary

This page records product-side evidence boundaries only. It does not modify the authority record, source registry, manifests, validators, tests, workflow state, candidate state, release records, publication pointers, or platform-owned controls. It does not infer or authorize a defect, approval, workflow transition, publication, promotion, release-pointer action, rollback, scheduling action, activation, or live-directory write. Keep unresolved facts and candidate-only status unchanged until separately governed evidence and authority exist.

Acceptance checks

- **AC-1 — Supported decision:** The reader can identify the declared and observed preflight routes, preserve their exact command and result fields, and state the one unresolved candidate-only decision without treating a declaration as execution proof.
- **AC-2 — Uncovered decision aid:** The reader can name the recurrence, causality, product-impact, affected-work, and owning-route evidence still required to distinguish an environment blocker from a product repair route, without inventing any missing fact.
- **AC-3 — Read-only boundary:** The reader can confirm that this page changes no workflow, authority, release, publication, pointer, scheduling, rollback, platform-attestation, or live-directory state and grants no transition authority.

Opportunity Audit

Audience: owner, onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Audit Scope

Authority is sealed before this conclusion: `sources/authority.json` records authority revision `2026-08-08`, lists `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and sets `employee_may_modify` to `false`; `registries/sources.json` resolves those IDs to public-safe records. The selected workstream is Internal Knowledge, the audience is Lee and current company operators, and the intended outcome is engineering velocity. The success hypothesis is that one explicit, candidate-only workflow slice, tied to sealed inputs and reproducible reader checks, will reduce ambiguity and handoff delay when operators prepare the next governed workflow. This audit is read-only and records a product opportunity, not authority to implement or execute a governed transition.

Evidence Inventory

The evidence examined for the conclusion is the bounded inventory named by the audit contract: `sources/authority.json`, `registries/sources.json`, `sprint-plan.md`, `context.md`, `WHERE_AM_I.md`, and `questions-for-the-human.md`. The contract at `artifacts/opportunity-audit/contract.md`, the page identity in `registries/pages.json`, and the prior contents of this page were checked only as audit-boundary and preservation controls; they are not evidence for the backlog conclusion. The authority file identifies the three source IDs and their governing-repository references; the source registry records each as public-safe. The sprint and context files show that the smoke-runner manifest contract slice is closed while the next governed workflow remains open. The location and question records identify a separate human-gate decision. This inventory does not establish a detailed `playbooks/` design, platform execution, approval, or publication outcome, so none is inferred here.

Active Backlog Assessment

The active backlog is not exhausted. `sprint-plan.md` marks the first proof and the smoke-runner manifest contract slice complete, while leaving both the next meaningful governed workflow under `playbooks/` and the work to make that workflow explicit enough for unattended Agent-Orch operation open. `context.md` and `WHERE_AM_I.md` agree that the next milestone is to hand retained product evidence to the human gate and then choose a bounded workflow. The human question ledger separately keeps human approval and controlled-pilot authorization open. Together these records support preparation of one narrow workflow slice, but they do not identify permission to execute it or enough detail for a broader implementation recommendation.

Recommended Next Step

Recommend exactly one bounded next step: author a candidate-only Internal Knowledge playbook outline for the next governed workflow, limited to its purpose, named repository inputs, reader journeys, deterministic commands, expected product observations, and the escalation and publication boundary. Its success condition should be that Lee or a current company operator can repeat the bounded inspection and identify the next human decision without guessing authority or platform state. This recommendation is supported by the two open sprint items and the next-milestone notes above. It does not authorize writing that playbook in this audit, running the workflow, approving a candidate, scheduling work, or changing any protected control.

Authority and Publication Boundary

The separate governing repository owns the employee charter, authority contract, routing mandate, validator authority, activation rules, and publication policy. This product audit owns only the conclusion supported by the listed public-safe product evidence; evaluator or reader observations do not become platform attestations. The human question ledger records a fresh commissioning decision at a human gate and a candidate-only state without a publication pointer, but it does not supply approval for this audit or for the recommended future slice. No authority transfer, route selection, validator attestation, repository identity, human approval, promotion, rollback, scheduling, publication, pointer creation, or live-directory write is asserted or performed here. The recommendation therefore remains candidate-only and reversible until the separately governed human decision is made.

Page Registry Synchronization

Audience: onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

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.

Pre-authoring Reader-Journey Contract Preflight

Audience: auto-orch workflow authors, auto-orch workflow operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Pre-authoring Reader-Journey Contract Preflight

Purpose

Run this preflight before an auto-orch workflow author drafts semantic content or an operator lets a governed execution consume budget. It turns the sealed authority record and immutable reader-journey manifest into a small, actionable contract decision: either every required fact, input, command, and reference is current and the workflow may proceed, or the result names the exact unresolved field or path that must be repaired. The result is a product-side conclusion; it does not change authority, certify a platform route or validator, or grant approval, activation, promotion, publication, scheduling, rollback, or live directory authority.

Inputs Checked

Start with `artifacts/preflight/source-authority-seal.json`. Confirm its sealed authority revision, authority-listed source IDs, and the permitted public-safe reference metadata for each source in `registries/sources.json`. The current source IDs are `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`; their locator, revision, digest, authority class, and disclosure are inputs to the comparison, not prose to be inferred. Then read the immutable `artifacts/preflight/pre-authoring-reader-journeys.json`: its contract path, four journeys, acceptance criteria, source references, traces, expected results, and `command_allowlist` define what this preflight must cover.

Resolve every repository-relative reference named by those records. That includes `sources/authority.json`, `registries/sources.json`, this page at `content/public/pre-authoring-reader-journey-preflight.md`, and the declared product-contract command target `tests/test_product_contract.py`. Check that the page metadata and its single `registries/pages.json` record agree on page ID, title, visibility, audiences, reading modes, claim references, and source references. Treat the manifest as immutable and do not use `build/`, `releases/`, `publication/`, private governance material, or generated output as authority or as a substitute for a missing input.

Actionable Errors

Report an error with its category, repository-relative location, observed value, expected value, and bounded repair target. For authority errors, name the source ID and field when a locator, revision, digest, authority class, or `public-safe` disclosure is missing, stale, conflicting, or disallowed; stop and preserve the gap rather than guessing provenance. For input-grounding errors, name the missing or out-of-scope path, journey, claim, or expected result and do not invent a replacement. The manifest, its journeys, and its traces are not repaired to make a result pass.

For command errors, identify the claimed invocation and the allowlist entry it fails to match. Reject a command that is absent from the JSON `command_allowlist`, or whose contract, manifest, test, evidence, or other repository-relative target no longer resolves. For stale-reference errors, show the old or unresolved page ID, path, source ID, acceptance check, or manifest link and stop before execution. These messages should let the authorized producer correct one bounded input or page/registry mapping; they must not require an operator to diagnose the cause from a vague failure or to use a newly generated fact as authority.

Valid Contract Path

First parse the two immutable preflight records with the exact declared inspection commands: `python3 -m json.tool artifacts/preflight/source-authority-seal.json` and `python3 -m json.tool artifacts/preflight/pre-authoring-reader-journeys.json`. Next compare the sealed sources and all journey references with the in-scope repository paths, including the page header and matching registry record. Then check that the product-contract evaluator invocation is compatible with the manifest's allowlist; the declared bounded check is `python3 -m pytest -p no:cacheprovider tests/test_product_contract.py -q`. Do not silently broaden a command, substitute a stale path, or treat an unlisted invocation as reproducible.

When all comparisons resolve, record a clear `PASS` that states authority, inputs, command compatibility, and reference freshness were checked. That pass may be consumed by the next governed workflow step without a human operator reverse-engineering the failure or authoring an explanation by hand. The pass keeps the immutable manifest unchanged, preserves the independent evaluation boundary, and remains candidate-only product evidence. If any comparison fails, record the actionable error and stop before budget-consuming execution or semantic interpretation; a valid contract is the only path to proceed.

Acceptance Checks

1. **AC-1 — Authority validation:** The preflight validates the sealed authority references and reports missing, stale, conflicting, or disallowed authority as an actionable error instead of inferring provenance. It compares the authority-listed source IDs with their registered public-safe metadata and preserves unresolved gaps.
2. **AC-2 — Input grounding:** The preflight grounds every journey, input, and material claim in the declared repository-relative inputs and reports missing or unsupported inputs without inventing replacements. Each required path, journey goal, expected result, contract reference, and page/registry mapping is checked before execution.
3. **AC-3 — Allowlist and stale-reference detection:** The preflight verifies that each claimed evaluator command is compatible with the JSON `command_allowlist` and detects stale or unresolved contract, input, and manifest references before execution. An implausible-looking command or path is not accepted without a current, bounded match.
4. **AC-4 — Valid contract proceeds:** A valid contract receives a clear pass and can proceed to governed execution without manual diagnosis, while the immutable manifest, independent evaluation boundary, and candidate-only boundary remain intact. The pass grants no platform-owned authority or publication effect.

Preflight Delta, Queue Age, Blocker, and Ownership Analysis

Audience: agent-orch maintainers, Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Preflight Delta, Queue Age, Blocker, and Ownership Analysis

This candidate-only analysis is for agent-orch maintainers reviewing the selected Internal Knowledge workstream. It compares exactly the three queue records declared by the product inputs and keeps product observations, platform-owned evaluation evidence, and unresolved ownership distinct. It does not schedule work, mutate queues, publish, promote, roll back, activate, change a release pointer, or write a live directory.

Preflight Delta

The declared product records contain exactly three queues: `internal` / Internal Knowledge, `onboarding` / Employee Onboarding, and `customer` / Customer Education. Internal Knowledge remains the selected oldest underserved eligible queue with recorded age `3` and age rank `3`; Employee Onboarding records age `2` and age rank `2`; Customer Education records age `1` and age rank `1`. The recorded relative age differences are therefore one age-value unit from internal to onboarding and two from internal to customer, with onboarding one unit older than customer. These are comparisons of supplied values only: no time unit, timestamp, threshold, freshness window, or priority is inferred.

Nothing in this step changes either declared product record. The selection receipt still says all three queues are eligible in scheduling tier `1`, only Internal Knowledge is underserved, and Internal Knowledge is selected before item-level impact or feasibility. The new analysis page, its page-registry entry, its reader-journey manifest, and the deterministic projection are additional handoff outputs; they are not a queue update, scheduling command, or new queue authority. The queue registry also does not establish an independent queue-source identity or revision, so that provenance remains unresolved.

Queue-Age Record

The queue-age record is reproduced without assigning a meaning to the numeric unit. Internal Knowledge has `age: 3`, `age_rank: 3`, `eligible: true`, `underserved: true`, and `scheduling_tier: 1`; its receipt reason says it is selected as the oldest eligible underserved workstream in the highest tier. Employee Onboarding has `age: 2`, `age_rank: 2`, `eligible: true`, `underserved: false`, and `scheduling_tier: 1`; it remains visible but is not selected because the ordered rule excludes a non-underserved queue before item-level scoring. Customer Education has `age: 1`, `age_rank: 1`, `eligible: true`, `underserved: false`, and `scheduling_tier: 1`; it remains visible for the same stated reason.

The difference between a recorded age value and a live queue-age fact is material. The inputs provide no queue-entered timestamps, clock, unit, freshness result, age threshold, or independent queue-source authority. The analysis therefore records values and ranks, not elapsed time or operational priority. No schedule follows from the comparison, and the two non-selected queues are not described as unimportant, infeasible, permanently excluded, or clear of failures.

Failure-Coverage Matrix

Recent-failure coverage is an explicit omission for all three queues. No declared input supplies a per-queue failure sample, recent-failure window, timestamp, retry count, recurrence measure, affected claim, or affected work item. The omission does not mean that a queue is failing or that it is clear; it means the evidence needed to make either product conclusion is unavailable in this bounded record.

Queue	Declared age / selection record	Recent-failure coverage	Owner route	Next action	Stop condition
`internal` / Internal Knowledge	`3` / rank `3`; eligible, underserved, tier `1`, selected	Missing; no per-queue sample, recurrence, retry, timestamp, or affected work is declared	Agent-orch maintainer or the evidence-producing queue/evaluation route; no individual is named	Request a governed recent-failure record for Internal Knowledge while preserving the supplied selection fields	Stop at candidate-only analysis if coverage or source authority is absent or contradictory; do not infer a failure or schedule work
`onboarding` / Employee Onboarding	`2` / rank `2`; eligible, not underserved, tier `1`, visible comparison	Missing; no per-queue sample, recurrence, retry, timestamp, or affected work is declared	Agent-orch maintainer or the evidence-producing queue/evaluation route; no individual is named	Request the same governed coverage record for Employee Onboarding rather than treating deferral as a failure result	Stop if coverage remains unavailable; do not infer that non-selection means clear, failing, or permanently deferred
`customer` / Customer Education	`1` / rank `1`; eligible, not underserved, tier `1`, visible comparison	Missing; no per-queue sample, recurrence, retry, timestamp, or affected work is declared	Agent-orch maintainer or the evidence-producing queue/evaluation route; no individual is named	Request the same governed coverage record for Customer Education and retain its comparison position	Stop if coverage remains unavailable; do not infer a customer-queue failure state, priority, or schedule

The declared failure evidence is separate from the matrix. It records one attempt of `journey.resolve-reader-journey-harness-decision` using the exact command `python3 tests/check_documentation_smoke.py`, with exit status `1` and observed output `Operation not permitted` because the evaluator environment denied localhost socket access. This is an environment-specific evaluation blocker only. It does not establish a recurring product harness defect, retry count, affected claim, affected work item, or repair target, and it is not a per-queue recent-failure sample.

Blocker Ownership

The localhost permission observation belongs to the platform-owned evaluator/environment route. The declared inputs do not name an individual platform owner, so owner identity is unresolved and must remain so. The product page author does not own a product repair for that observation. The missing recent-failure coverage for `internal`, `onboarding`, and `customer` belongs to the agent-orch maintainer or the evidence-producing queue/evaluation route that can supply a governed record; that is a role-level route, not an invented person or independent queue-source identity.

Unresolved finding	Owner route	Next action	Explicit stop condition
Environment-specific localhost permission blocker	Platform-owned evaluator/environment route; individual identity unresolved	Request a permitted rerun of the exact journey and preserve its command, status, output, and recurrence fields	Stop without calling it a recurring product defect or naming a repair target
Missing recent-failure coverage for all three queues	Agent-orch maintainer or evidence-producing queue/evaluation route; individual identity unresolved	Request one governed record with per-queue samples, retries, recurrence, timestamps, source references, and affected work	Stop with omission if any required coverage is absent, stale, or contradictory
Independent queue-source identity and revision not established	Agent-orch maintainer or evidence-producing queue/evaluation route	Request an authoritative queue-source ID and revision before strengthening the comparison	Stop without assigning queue authority, freshness, threshold, or priority
Product defect or repair target not established	Analysis author preserves the unset status	Wait for product-specific governed feedback that names affected product work	Stop all product repair work until a defect and affected work are established

The independent queue-source identity and revision are also unresolved. The maintainer route owns the next request for that provenance, and the stop condition is to withhold any stronger queue authority, freshness claim, threshold, or priority until an authoritative record supplies it. Product repair status is unresolved in a narrower sense: no product defect or repair target is established. The analysis author owns preserving that status, the next action is to wait for product-specific governed feedback, and the stop condition is no repair work until such feedback names a product issue and affected work. A later product repair, if actually established, remains limited to this analysis page, its synchronized registry entry, and its journey manifest under the declared contract.

The ownership boundary does not rewrite the failed observation. A later permitted evaluator may rerun the exact journey in an environment with the required reader service and socket access, but that follow-up remains platform-owned evidence. It may add recurrence, source references, affected work, and an owner; it may not be backfilled by this page or converted into a platform attestation by a product conclusion.

Next Actions and Stop Conditions

The smallest permitted next action is to request one governed follow-up evaluation that preserves the exact journey ID, allowlisted command, exit status, observed output, retry or recurrence evidence, source references, affected work item, and recent-failure coverage for each of the three compared queues. Keep the candidate parked and preserve the current selection receipt, including the age values and ranks. Ask the evidence-producing queue or evaluation route to identify its authoritative queue-source identity and revision, and ask the platform-owned evaluator/environment route to retain the localhost-denied observation rather than replacing it.

Stop the analysis at the candidate-only boundary whenever recent-failure coverage, recurrence, affected work, or authoritative ownership is missing, stale, or contradictory. Also stop if a reader attempts to turn age values into a time threshold, a non-selected queue into a failure or clearance result, or the environment permission error into a recurring product defect. No queue mutation, schedule, approval, publication, promotion, rollback, activation, release-pointer change, live-directory write, or guessed repair prose is permitted by this record. The outcome remains unresolved coverage with no product repair target and no transition until governed evidence changes that conclusion.

Acceptance Checks

- **AC-1 — Delta:** The reader verifies that the comparison contains exactly Internal Knowledge, Employee Onboarding, and Customer Education, records age values and ranks `3`, `2`, and `1`, and states relative deltas without inventing an age unit or queue authority.
- **AC-2 — Queue age:** The reader retains eligibility, underserved status, and tier for all three queues, confirms Internal Knowledge as the selected oldest underserved eligible queue, and does not infer a time threshold, freshness result, or schedule.
- **AC-3 — Failure coverage:** The reader preserves the exact one-attempt journey ID, command, exit status `1`, and `Operation not permitted` observation as an environment-specific evaluation blocker, while recording per-queue recent-failure coverage, recurrence, retries, timestamps, affected work, and samples as missing.
- **AC-4 — Ownership:** The reader routes the environment observation to the platform-owned evaluator/environment route and missing queue coverage to the agent-orch maintainer or evidence-producing route, while stating that no named individual or independent queue-source owner is established.
- **AC-5 — Next action:** The reader recommends only a permitted follow-up evaluation capturing the missing evidence fields, preserves the candidate-only handoff, and leaves any product repair target unset.
- **AC-6 — Stop condition:** The reader stops when required coverage, recurrence, affected work, or ownership evidence is absent or contradictory and confirms that no queue mutation, scheduling, publication, promotion, rollback, activation, pointer, or live-directory action follows.

Publication Candidate Packet Contract

Audience: owner, independent reviewers, governed-run operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Publication Candidate Packet Contract

This contract defines a bounded product-side packet for reviewing a possible publication candidate. It does not create a candidate, select a release, or attest that a transition occurred. A missing, stale, conflicting, or unprovided fact remains unresolved.

Authority boundary

The sealed authority input is `sources/authority.json`. It records revision `2026-08-08`, lists exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and sets `employee_may_modify` to `false`. The packet records those references without copying private governance material or treating the authority record as approval to publish.

The approved-source boundary in `sources/approved/README.md` says that logical references and digests belong to the source registry and that missing or stale authority blocks governed delivery. The packet therefore does not fill absent source or evidence details with generated prose.

Evidence and selection boundary

The packet must keep declarations separate from observed evidence. A command allowlist is not proof that a command ran, and product-side validation, reader, or review evidence is not a platform-owned attestation. This step declares no trusted artifact inputs, so candidate identity, comparison basis, reader evidence, independent review evidence, and readiness remain unresolved unless separately supplied by an authoritative handoff.

Candidate-only boundary

The packet remains candidate-only. It does not authorize or infer approval, promotion, publication, scheduling, rollback, release-pointer changes, or a live-directory write. The repair is limited to this packet's content, contract, page registry entry, model, and journey manifest; the sealed `sources/` authority remains read-only.

Acceptance checks

- **AC-1 — Sealed authority trace:** The reader can identify the authority path, revision, exact three source IDs, and `employee_may_modify: false` without copying private governance material or modifying the authority.
- **AC-2 — Bounded evidence and selection:** The reader can distinguish declared evidence requirements from observed evidence and leaves candidate selection unresolved when the required candidate, validation, reader, or review artifacts are not supplied.
- **AC-3 — Candidate-only handoff:** The reader can state that this packet is product-side review material only and does not authorize or infer approval, promotion, publication, scheduling, rollback, release-pointer changes, or a live-directory write.

Publication Reader Journey

Audience: prospective-client, owner | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Publication Reader Journey

A qualified prospective or current Governed Execution Engine customer begins at the public engine overview, names one bounded documentation use case, and uses this page to find the capability being evaluated: a traceable, public-safe way to read governed documentation evidence. This is a reader route for fit evaluation, not a performance, compliance, availability, or customer-outcome promise. The reader keeps unsupported, stale, or conflicting facts visible while deciding whether the use case merits a scoped adoption conversation.

Evidence and Authority

Start with the sealed `sources/authority.json`. Its authority revision and `source_files` identify `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`. Follow those IDs in the `registries/sources.json` source registry and check each recorded locator, revision, digest, and `public-safe` disclosure. The `approved-source boundary` says that a missing or stale authority reference blocks the governed delivery; it does not permit the reader to fill a gap with plausible prose. The reader may record a product conclusion about the wording and evidence, while platform-owned validator, identity, route, approval, and activation attestations remain outside this page.

The publication boundary is described by the product release contract (`docs/release-contract.md`): a candidate is a local, reversible record for review, not proof that a live transition happened. A customer may inspect a candidate's evidence and readiness facts, but must not interpret that inspection as approval, promotion, publication, scheduling, rollback, creation or change of a release pointer, a live release, or a live directory. If the available authority does not establish one of those outcomes, the correct reader result is an explicit unresolved gap.

For version and rollback interpretation, compare the immutable candidate record's previous-version identity, rollback-target identity, rollback availability, and any release-blocking fact. A missing, stale, or conflicting fact remains unresolved; do not infer a previous release, a rollback target, or rollback authority from candidate evidence. The candidate record is evidence to inspect, not a platform attestation.

To reproduce the bounded evidence trail, follow the registered sources and compare `build/index.json` with the deterministic content-validation, projection, and smoke evidence named for the candidate. Those checks support a product conclusion about the documented evidence; they do not create a release pointer, promote a candidate, publish content, schedule or roll back a release, or write a live directory.

Next Adoption Decision

After completing the source trace and candidate-boundary check, the next adoption decision is whether one named, bounded documentation use case merits a scoped evaluation conversation. Compare the use case's required claims and evidence with the sealed source records, note every missing, stale, or conflicting fact, and decide whether the evidence is sufficient to continue fit review. A reversible next step is to refine the use-case question or ask the responsible authority to resolve a documented gap. It is not to treat a candidate as a live release, create a publication pointer, or perform any promotion, publication, scheduling, rollback, or live-directory write. This journey records customer-fit evidence only and leaves platform attestations to the platform-owned gate.

Publication Reader-Journey Gaps

Audience: owner, independent reviewers, governed-run operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

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.

Publication-Candidate Packet

Audience: owner, independent reviewers, governed-run operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Publication-Candidate Packet

This is Lee's product-side publication-readiness report for the strongest sealed candidate. The selection record identifies `repair-8ab0ca9952d8` as the newest sealed candidate with two-page coverage, passed product validation, passed reader evidence with no findings, and a prior independent review with no findings. It is evidence for a human decision, not approval, promotion, publication, serving, rollback, or a live-directory action. The report keeps the candidate-only metadata and boundary intact.

The current sealed source boundary is `sources/authority.json`, revision `2026-08-08`, SHA-256 `973d42700fcf5046c8c1b8b408337987537e04d8eec4c742574fb30f7a10c3b6`, with exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`; `employee_may_modify` is `false`. The three source records are registered `public-safe`. Private governance material is not copied into this packet.

Verified Projection

The strongest sealed release is `repair-8ab0ca9952d8`, selected by `releases/publication-readiness-candidate.json` and described by `releases/publication-readiness-report.md`. Its immutable release manifest is `releases/repair-8ab0ca9952d8/release.json`; it records `status: candidate`, `publication_status: candidate-only`, `promotion_ready: true`, `previous_release: null`, `rollback_target: null`, and `rollback_available: false`. `promotion_ready` is a product-side candidate fact and is not approval or permission to serve.

The exact immutable product serving artifact available in the sealed release evidence is `releases/repair-8ab0ca9952d8/projection/index.json`, SHA-256 `26785c2bd526d50389afed09571debdb1235a64354370613c845c0837166977e`. It is an immutable projection index, not proof that a complete externally managed webroot bundle or nginx configuration is available. Its identity is content revision `881f1b6771dc612d0f3472306cb6d3797fb230846a5f69b08718048b511a80af`, release authority digest `5e8f4793ce2e526a73f0065e0c045849ff8610e934d83e3307a0c1af52d9a549`, renderer `renderer-v0`, and theme `theme-v0`. It names claim `claim.governed-documentation-product`, models `models/engine-overview.json` and `models/no-additional-work-reader-journeys.json`, and exactly these public pages:

- `page.governed-execution-engine-overview` at `content/public/engine-overview.md`, with body SHA-256 `51ef85f994378f69fdcf7ab44b69e779ed6b3491ef5eaa5d94fbb09adab1850b`.
- `page.no-additional-work-cycle` at `content/no-additional-work-cycle.md`, with body SHA-256 `8cf20809d7868d059ed8b8f9629b9caa140ce4572c0dd432ab25db20dc42e8b3`.

The immutable `validation.json` records `passed: true` for seven product checks: schema and references, model and craft schemas, internal links, protected and generated paths, reader journeys, public safety, and reproducible projection. The selected release's reader record marks `journey.owner-overview` passed, with command `python3 tools/validate_content.py --journey journey.owner-overview`, exit code `0`, and no findings. Its “Ready for controlled pilot” recommendation is product-side evidence only. The attached review record is a prior sealed pass with no findings; it is not a fresh independent review for this report.

The selected release snapshot carries authority revision `2026-08-04` and the older digest above, while the current sealed authority is revision `2026-08-08` with the current SHA-256 above. The workspace `build/index.json` also has a different newer content revision, `b82ec59383bbffd712dbdda59f75468594044d84e9e7a43d1e3d82d5bf01261e`. Neither current state is substituted for the named immutable artifact.

The preserved validator evidence for this handoff was not rerun. The exact preserved command was:

```
PYTHONDONTWRITEBYTECODE=1 python3 -c "import hashlib; from pathlib import Path; assert hashlib.sha256(Path('sources/a
```

It exited `0`, was marked passed, took `0.083659` seconds, and has evidence hash `b451aeac25a532f1f5ee712593e1ed66861e5504f6e33581b6c3a6ed92b200e6`. The preserved record reports empty stdout and stderr; no result counts are claimed.

Embarrassment-Risk Gaps

These are all remaining risks visible in the supplied evidence; none is resolved by readiness wording or by the presence of an immutable candidate:

1. Lee's human readiness decision is open. The product report can recommend review, but it cannot supply human approval. Controlled-pilot authorization is a later and separate decision.
2. No authority for the externally managed serving boundary is present. The workspace contains no authorized webroot path, nginx configuration or site identifier, serving owner, permitted staging/switch mechanism, health check, or rollback runbook. No such path, configuration, or command may be inferred.
3. The only exact immutable product serving artifact named by the sealed release evidence is the projection index above. The evidence does not prove that it is a complete renderable webroot bundle, so a human must decide its sufficiency or provide a separately authorized immutable bundle manifest.
4. The selected release's authority snapshot is older than the current sealed authority. The preserved authority hash check establishes the current file's digest; it does not re-verify the selected release against current authority.
5. The selected reader evidence covers only `journey.owner-overview`. The three journeys in the dedicated report manifest are reader-contract requirements, not executions performed by this authoring step.
6. The separate platform-owned journey `journey.resolve-reader-journey-harness-decision` has one recorded attempt: `python3 tests/check_documentation_smoke.py` exited `1` with observed output `Operation not permitted` because the evaluator environment denied localhost socket access. This establishes an environment-specific evaluation blocker, not a recurring product harness defect, retry count, affected claim, or repair target. A permitted follow-up must preserve the journey ID, exact command, exit status, output, recurrence or retry evidence, source references, and affected work item before a defect is named.
7. Platform-owned route, validator-authority, repository-identity, execution, evidence-chain, terminal, approval, and serving attestations are not established by this product packet.
8. The selected candidate records no approved previous release and no rollback target. Older candidates are comparison evidence, not approved prior versions. A reversible serving action cannot honestly name a rollback target until a human-authorized operator records one.
9. `publication/current-release.json` is absent, and the selected immutable snapshot differs from the current workspace projection. The candidate must not be represented as the current release, and the current projection must not overwrite or silently replace the sealed snapshot.
10. The selected release's `promotion_ready: true`, reader recommendation, and prior review pass are candidate-evidence facts. They do not establish publication approval, a serving action, a live-directory write, or a fresh independent review for this report.

Publication Boundary

This report and `repair-8ab0ca9952d8` remain candidate-only product material. The local conceptual boundary is `publication/current-release.json`, which is absent and was not created or changed. No approval, promotion, publication, scheduling, activation, rollback, pointer mutation, or live-directory write was performed or is claimed. In particular, the exact immutable projection index is identified for Lee's review; it has not been copied to, mounted at, or served from any external webroot. The report cannot establish that serving would be safe until the missing external boundary authority and artifact sufficiency decision are supplied.

The human decision needed is bounded: Lee must decide whether to authorize the reversible serving plan after an authorized operator supplies the exact externally managed webroot path, nginx site/config identity, permitted change mechanism, health-check evidence, and rollback target. Lee must also decide whether the named immutable projection index is sufficient or whether a separately sealed complete serving bundle is required. Nothing in this packet grants that authority or treats the product recommendation as publication.

Reversible Serving Plan

The following is a conditional plan for the existing externally managed boundary, not an execution record. First, the authorized operator supplies the missing webroot and nginx boundary authority, identifies the current serving target and a recoverable rollback target, and records the permitted staging and switch procedure. No path or nginx command is invented here. Second, Lee reviews the exact artifact `releases/repair-`

`8ab0ca9952d8/projection/index.json` and its SHA-256, and decides whether that projection index is the approved serving artifact or whether a separate immutable bundle must be supplied.

Only after those decisions should the authorized operator stage the approved immutable artifact without modifying its release directory, capture the before-state and staged hash, and use the existing approved boundary procedure to make a reversible switch. The operator then performs the human-approved health check and records the after-state. If the hash, artifact completeness, health check, or boundary authorization does not match, the operator stops and leaves the candidate parked. If service is unhealthy, the operator restores the previously recorded approved target; because this candidate records no prior approved release, no rollback target may be invented. Lee's reversible default until all of this authority is present is no serving action, no pointer change, and the candidate remaining parked for review.

Queue Input Feasibility Receipt

Audience: workflow maintainers, workflow operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Queue Input Feasibility Receipt

Purpose

This is a read-only, candidate-only Internal Knowledge handoff for workflow maintainers and workflow operators. It checks whether the fixed inputs for the already selected `internal` queue form a usable product contract. The upstream selection remains the Workstream Queue Selection Receipt: `internal` is selected, while `onboarding` and `customer` remain visible comparators. Feasibility is downstream of that selection and cannot replace it, schedule work, mutate queue state, or turn a product conclusion into a platform attestation. The receipt is public-safe because its page claims are empty and its material source references are the three sealed, public-safe logical IDs.

Fixed Representative Fixtures

`registries/queue-input-feasibility-fixtures.json` is the sole fixed fixture inventory for this receipt. It contains the complete three-row upstream set (`internal` , `onboarding` , and `customer`), one selected-queue case with ready sealed authority and a separately recorded freshness gap, and explicit `missing` , `stale` , `conflicting` , and `non-public-safe` cases. The latter are test-shaped blocked inputs, not observations about a live queue and not permission to repair authority. Each row retains age, age rank, eligibility, underserved status, scheduling tier, audience, intended outcome, selection reason, source references, freshness fields, and balance-window membership. The fixture file also records the upstream page and candidate identities, required artifacts, standard acceptance IDs, and command allowlist so another reader can derive the same result without inventing a queue source identity or revision.

Eligibility Checks

An input is eligible for a product feasibility conclusion only when it has a valid contract, uses the standard acceptance IDs AC-1 through AC-5, has complete non-exploratory journey coverage, and names only non-empty commands from the fixture allowlist. The required artifacts are the sealed authority, registered sources, upstream selection page, upstream candidate record, this page and its registry entry, the fixture inventory, the journey manifest, and the deterministic projection. The sealed authority is ready here at revision `2026-08-08` with exactly the three registered public-safe sources and `employee_may_modify: false` . The fixture has a complete balance window for all three queue IDs, but its independent queue-source identity and revision are unestablished; therefore queue-input freshness is explicitly `blocked` , not guessed from queue age. Missing, stale, conflicting, or non-public-safe inputs fail closed with an owner-visible finding.

Derivation Record

The deterministic derivation first checks the fixture schema and exact acceptance-ID set, then matches the upstream receipt identity and all three queue rows, then checks required artifacts, journey traces, command strings, sealed authority, source disclosure, freshness, and balance-window evidence. It next confirms that `internal` is still the upstream selected ID and that feasibility has not been used as a tie breaker. The authority readiness result is `ready` ; the balance-window result is `ready` ; the queue-source freshness result is `blocked` because no independent queue-source identity or revision is established by the candidate; and the product feasibility conclusion is `blocked-pending-freshness-evidence` . The derivation is deterministic and fixture-backed, while the unresolved platform reader follow-up remains platform-owned evidence rather than a product defect or attestation.

Read-Only Boundary

This receipt describes product evidence only. It does not schedule work, mutate queue state, alter the selected workstream, create or change a release pointer, promote, publish, roll back, activate, or write a live directory. It does not claim that a reader journey, independent review, route, identity, validator authority,

evidence chain, approval, or platform gate has executed. The preserved validator record is authoritative evidence supplied for the handoff; it is not replaced or rerun here. A missing, stale, conflicting, or non-public-safe input remains visibly `blocked` with its reason and owner. In particular, the environment-specific reader result is not converted into a recurring product harness defect, and no platform-owned fact is inferred from the product fixture or deterministic build.

Acceptance Checks

- **AC-1 — Authority:** Confirm the exact sealed authority revision, three logical source IDs, immutable employee flag, and public-safe source records; missing, stale, conflicting, or non-public-safe authority is blocked.
- **AC-2 — Contract and scope:** Confirm a valid candidate contract, complete required artifacts, the upstream receipt identity, and all three fixed queue inputs, with `internal` remaining selected before feasibility is considered.
- **AC-3 — Journey coverage:** Confirm standard IDs AC-1 through AC-5, complete non-exploratory `traces_to` coverage, natural-language reader journeys, and non-empty commands that occur in the JSON allowlist.
- **AC-4 — Evidence and derivation:** Confirm ready sealed authority, explicit freshness evidence, complete balance-window evidence, and deterministic fixture-only derivation; unresolved queue-source freshness stays blocked.
- **AC-5 — Boundary:** Confirm the result is a read-only product conclusion with open findings preserved and no schedule, mutation, platform attestation, approval, promotion, publication, rollback, pointer, or live directory effect.

Reader Journey Gate Escalation

Audience: Lee, current company operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Reader Journey Gate Escalation

Purpose

This page is the Internal Knowledge workstream's bounded operator guidance for Lee and current company operators. Use it when a required reader journey may be impossible to satisfy, and use the threshold narrowly: escalation is appropriate only after the available product repair route has been exhausted and the remaining failure is a proven authority or platform capability gap. This page records a product conclusion and a decision request; it does not grant authority, change governance, or imply that a publication transition has occurred.

Prove Unsatisfiability

Name the exact journey and its required observable result before drawing a conclusion. Read `sources/authority.json` first, then match every listed source ID in `registries/sources.json` by locator, revision, digest, and `public-safe` disclosure. Record the bounded repairs attempted, the exact product evidence produced by each attempt, and the unresolved authority or platform blocker. The proof is complete only when it explains why product content, registration, source references, or projection work cannot repair that blocker and identifies the smallest decision or advice Lee must provide. An unresolved fact is evidence to retain, not permission to fill a gap with inference.

Do Not Escalate Repairable Defects

Treat malformed journey data, missing page registration, broken source references, insufficient section content, and stale deterministic projection as bounded product defects. Repair the affected content, registration, source reference, or projection; rebuild the projection; and rerun the declared product check, retaining the failed observation beside the later result. A passing repair is evidence about the product route only. Missing publication, promotion, rollback, or platform-attestation evidence is not by itself proof that the reader journey is unsatisfiable, and none of those boundaries may be crossed to manufacture an escalation result.

Evidence for Lee

The decision packet should contain the journey ID and goal, the source mapping, the attempted repair record, the remaining blocker, and the smallest proposed decision. The approved source mapping for this workstream is `source.naming-decision` at `factory:decisions.md#Governed Execution Documentation Steward naming and route`, revision `2026-08-02`, digest `7c42ccc9f454a81e2c119c14cdc3ad5ba30ef54d27fb6a8b371c7a640eb54dd2`, `source.documentation-boundary` at `product:architecture.md#Semantic source`, revision `2026-08-04`, digest `0094e6f534a04729008a3c85510899e5022cbd18a0f7ac9302b19f0c3ba29f76`, and `source.platform-delivery-contract` at `agent-orch:docs/delivery-profile-contract.md#content`, revision `agent-orch-main-e65a64fcf429`, digest `d466a6dca8420477f416a535c9c2bd76aa314fb5e67d72d405b12e6386a9573b`. Keep Lee's request separate from product conclusions: an evaluator may state whether the threshold is met, while platform-owned gate evidence and any authority decision remain with their owners. If the proof is incomplete, ask for nothing and record no escalation; retain the candidate-only boundary and all publication state unchanged.

Acceptance Checks

- **AC-1 — Prove the authority or platform blocker:** The record names the exact reader journey, preserves its required result and source mapping, and identifies the unresolved authority or platform capability gap together with evidence that the gap cannot be repaired within the product content surface. It

does not treat an absent attestation or an unperformed publication transition as proof of unsatisfiability.

- **AC-2 — Rule out bounded repairable defects:** The record shows that malformed journey data, missing registration, broken source references, insufficient content, and stale projection were checked and repaired or otherwise excluded, with failed observations retained beside later product results. A repair result remains product evidence and is not promoted into platform authority.
- **AC-3 — Provide Lee a minimal evidence-backed decision request:** Only when AC-1 and AC-2 are satisfied, the packet asks Lee for the smallest named decision or advice needed to resolve the proven blocker, links that request to the retained evidence, and states the reversible no-escalation outcome when the proof is absent. The request implies no approval, promotion, publication, scheduling, rollback, pointer change, or live-directory write.

Reader-Journey Repair Decision Brief

Audience: owner, onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Reader-Journey Repair Decision Brief

Purpose

This brief gives auto-orch maintainers and sprint decision-makers a bounded, public-safe decision record for the Internal Knowledge reader journey. The sealed authority revision is `2026-08-08`, and the current defect inventory concludes `no_defect_established`; therefore no speculative repository repair is authorized. The brief separates product evidence from unresolved reader, environment, and platform facts, while keeping governance, routing, validator authority, activation, approval, publication, and live-directory controls outside this product-owned decision.

Defect Triage

The current triage result is **no defect established**. The dependency order is authority boundary, deterministic reader-contract shape and trace coverage, independent reader evidence, then environment or platform-owned facts. No deterministic contract failure is established. No reader-evidence failure is established, although next-run reader evidence remains unknown. No environment failure is established because no owning environment record is supplied. A missing, stale, conflicting, or unsupported fact stays unresolved; it is not a defect, a pass, or permission to infer a platform result. The three authority-listed logical sources are public-safe, and private governance material remains outside this conclusion.

Minimal Repair Paths

The smallest safe path now is reversible no-repair: leave product sources and governed controls unchanged until a later governed result supplies concrete validation feedback. If that result proves a product defect, repair only the earliest failed deterministic prerequisite—such as malformed structure, an unresolved reference, or missing acceptance trace—and retain the failed observation beside the later result. If deterministic checks pass but reader evidence is absent or contradictory, repair the bounded evidence procedure and preserve the discrepancy. If an environment or platform fact is missing, make no product repair and route the gap to its owning authority. Regenerate the projection from source after any permitted repair; never treat generated output as semantic source.

Lee Decisions

This brief asks Lee for no immediate repository mutation. The smallest human decisions that remain open are approval of a later readiness recommendation and authorization of a controlled pilot. Until Lee records an applicable decision, the reversible default is to keep the no-repair conclusion and any later candidate evidence parked and unchanged. Product readiness cannot substitute for approval, pilot authorization, route or identity attestation, validator authority, promotion, publication, scheduling, rollback, a pointer change, or a live-directory write. Any such transition must use its separately governed authority boundary.

Acceptance Checks

The maintainer should be able to verify the sealed authority revision and its three public-safe sources; distinguish deterministic, reader-evidence, and environment categories; and identify that `repair_target` is currently null. The next governed run should execute every non-exploratory journey once by manifest ID with an exact allowlisted command, recording steps, exit status, observed result, source references, and `traces_to`. Failed output and later results remain side by side. The product conclusion must remain separate from platform-owned attestations, and no acceptance check may imply Lee's approval or authorize promotion, publication, scheduling, rollback, a pointer, or a live-directory action.

Smoke-Runner Manifest Contract

Audience: owner, onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Smoke-Runner Manifest Contract

Overview

This page is the operator-facing contract for the one Internal Knowledge workstream serving Lee and current company operators. Its intended outcome is engineering velocity through a short, reproducible documentation smoke path; that is an intended product outcome, not a measured platform result. The sealed oracle is `tests/smoke_manifest.json`, whose current SHA-256 is `b53bea17c6a278817074fa8d87365d0ad4b329453dc8e1b003c180fa211938c1`. That manifest names the product oracle `tests/test_product_contract.py`, pinned at `76c6b6e1fcaa1e8e2851134c32db89e54d1dc8857f5240dea2d56dea4f75290c`. Operators use those pins, the exact command order below, and the journey manifest to work from sealed inputs instead of rediscovering sources. The page records product evidence only; it does not authorize deployment, publication, promotion, rollback, scheduling, pointer changes, or live-directory writes.

Validation Contract

Before semantic output is trusted, an operator must parse `tests/smoke_manifest.json`, verify its SHA-256 pin, and then follow the manifest's declared smoke sequence in order. The separate `tests/smoke_manifest.agent-orch.json` is an orchestration gate and cannot replace the product oracle or product-side evidence. The product oracle is read-only for this run: if its digest differs, stop and preserve the mismatch rather than changing the expected pin or rerunning an unsealed test.

Run the deterministic preflight in this order:

1. Parse and seal `tests/smoke_manifest.json` at `b53bea17c6a278817074fa8d87365d0ad4b329453dc8e1b003c180fa211938c1`.
2. Verify the pinned oracle digest for `tests/test_product_contract.py` at `76c6b6e1fcaa1e8e2851134c32db89e54d1dc8857f5240dea2d56dea4f75290c`.
3. Run the declared oracle exactly as `python3 -m pytest tests/test_product_contract.py -q` and retain its exit code and failure output as product evidence.
4. Seal `sources/authority.json` at `973d42700fcf5046c8c1b8b408337987537e04d8eec4c742574fb30f7a10c3b6` before interpreting or producing semantic output.

After that preflight, run the bounded projection command `python3 tools/build_projection.py --output /tmp/gee-smoke-index.json` and compare the temporary JSON structurally with `build/index.json`. A mismatch is a stale or unexplained generated projection and must be reported for an authorized producer; this page does not make generated output current by assertion and the operator run does not write a live directory.

The six acceptance obligations are:

- AC-1 seals the parseable smoke manifest before consumption and identifies the sealed product oracle.
- AC-2 generates a bounded projection and compares it with `build/index.json`, reporting any stale output rather than silently repairing it.
- AC-3 seals `sources/authority.json` before semantic interpretation; its required digest is the value stated above.
- AC-4 checks this page under the exact `Overview`, `Validation Contract`, and `Operator Runbook Journey` headings, with reader-useful content under each heading.
- AC-5 requires `journeys/user_journeys_manifest.json` to declare complete, safe, reproducible operator journeys and to cover AC-1 through AC-6.
- AC-6 requires an independent evaluator and an independent reviewer to keep product conclusions separate from platform-owned attestations and to cite an allowlisted deterministic check.

Each writable semantic output therefore needs its own content-bearing check: the page's headings and explanatory meaning are inspected, the journey manifest is parsed and its commands are replayable, and registry references resolve. A successful file-existence check or unrelated command is not enough. These are

product conclusions only; validator identity, routing, activation, publication, and other platform attestations remain owned by the governing repository and its platform evidence. A clean product smoke result therefore does not by itself establish approval, deployment, publication, or readiness.

These checks establish a reproducible product conclusion. They do not replace the governing repository's authority contract, routing mandate, validator authority, activation rules, or publication policy.

Operator Runbook Journey

Start at the repository root with the sealed paths named in the manifest. Use `journeys/user_journeys_manifest.json` as the runbook index: every non-exploratory journey has one exact command, one expected observable result, and a `traces_to` entry for at least one page acceptance criterion. Exercise the journeys in their listed order after the preflight: seal the manifest, confirm the bounded projection path, seal authority, confirm the page contract, confirm the journey bindings, and obtain the independent evidence boundary. The manifest's `command_allowlist` contains the complete command prefixes; copy a journey's `command` verbatim so a second evaluator can reproduce it without guessing arguments or discovering another source.

For diagnosis, stop at the first failed gate and preserve the exact command, exit code, stdout, stderr, and observed digest. A manifest parse or digest failure means the smoke oracle is not sealed: compare the file bytes with the documented digest and do not edit the manifest to satisfy the pin. An oracle digest failure means `tests/test_product_contract.py` is not the pinned test: retain the mismatch and do not treat a later test result as valid. A non-zero pytest result is a product-contract failure: retain the named failing test and its output, then report the failure without converting it into a platform finding. An authority mismatch means semantic interpretation must stop until the sealed authority input is reconciled. A projection mismatch means the bounded output and `build/index.json` differ; report the differing structure or hash and leave generated-output repair to an authorized producer. A page, journey, or review-boundary failure means the product evidence is incomplete, not that approval or publication has occurred. This record-and-stop pattern keeps the failure actionable while preserving the publication boundary.

Source Authority Orientation

Audience: auto-orch operators | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Source Authority Orientation

This Internal Knowledge orientation gives an auto-orch operator a repeatable, read-only path to the authoritative internal source references before interpreting product guidance. It identifies source records and permitted provenance metadata; it does not reproduce source corpora, create authority, or establish a workflow, evaluator, runtime, or transition result.

Authoritative Sources

Begin with `sources/authority.json`. Its sealed revision is `2026-08-08`, its `employee_may_modify` value is `false`, and its ordered `source_files` list is exactly `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`. That file establishes which logical source IDs belong to this orientation. It does not grant permission to edit the authority record or to copy the private material behind a locator.

Reconcile each ID with exactly one record in `registries/sources.json`. The registry exposes only the reference metadata allowed by the contract: authority class, locator, revision, digest, and disclosure. The complete sealed mapping is:

Source ID	Authority class	Locator	Revision	Digest
`source.naming-decision`	`canonical-platform-source`	`factory:decisions.md#Governed Execution Documentation Steward naming and route`	`2026-08-02`	`7c42ccc9f454a81e2c119c14cdc3ad5ba30`
`source.documentation-boundary`	`product-owned-public-safe-source`	`product:architecture.md#Semantic source`	`2026-08-04`	`0094e6f534a04729008a3c85510899e5022`
`source.platform-delivery-contract`	`canonical-platform-source`	`agent-orch:docs/delivery-profile-contract.md#content`	`agent-orch-main-e65a64fcf429`	`d466a6dca8420477f416a535c9c2bd76aa3`

The three records above are the whole source boundary for this page. A convenient Markdown page, `build/index.json`, generated prose, a candidate packet, an evaluator result, a runtime observation, or a command declaration cannot add a source or replace a missing registry record. If an ID, field, revision, digest, or disclosure value is absent, stale, conflicting, or not public-safe, preserve that fact as unresolved and stop source interpretation.

Orientation Workflow

Follow this order for every auto-orch handoff:

1. Read `sources/authority.json` first. Record the authority revision, the `employee_may_modify` flag, and the exact ordered logical source list. Do not add a source because it appears useful or because another page names it.
2. Read `registries/sources.json` and match every listed ID exactly once. Check the authority class, locator, revision, digest, and `public-safe` disclosure from the registered record. Treat these as provenance metadata, not as a summary of the referenced source contents.
3. Use `sources/approved/README.md` for the repository disclosure boundary and keep private governance material, private source corpora, and runtime evidence out of the orientation. A missing or conflicting reference stays visible as an unresolved gap; it is never repaired with plausible prose.

4. Use the synchronized reader journeys in `journeys/source-authority-orientation.json`: follow `journey.source-authority-seal` for the authority boundary, `journey.source-authority-registry-match` for the three registry records, and `journey.source-authority-orientation-boundary` for the read-only interpretation. A command in that manifest is an allowlisted procedure, not proof that the command ran or that an external transition occurred.
5. Before handing off, compare the page metadata with its single registry object and ensure the source references remain the exact sealed set. Keep source identity, product conclusions, candidate evidence, and any platform-owned attestation in their separate evidence boundaries.

This workflow ends with source orientation only. It does not approve, activate, promote, publish, schedule, roll back, change a release pointer, or write a live directory. Those controls are outside this page, and no state about them should be inferred from a synchronized page, generated projection, journey manifest, or successful product-side check.

Source-Identification Check

Use this check before treating an internal reference as authoritative. Confirm that the authority record says revision `2026-08-08`, `employee_may_modify` is `false`, and the source IDs appear in this exact order: `source.naming-decision`, `source.documentation-boundary`, `source.platform-delivery-contract`. Then confirm that the registry contains one and only one record for each ID, with a non-empty authority class, locator, revision, and digest, and with disclosure equal to `public-safe`. Confirm that no extra registry record has been folded into the sealed source list.

The check passes as source orientation only when all of those comparisons resolve and the reader can explain the difference between authority and evidence. `sources/authority.json` supplies source identity; the registry supplies permitted provenance metadata; the contract and journey manifest define how to read that boundary. A page file or `build/index.json` is a generated or semantic product artifact, not an authority source. A candidate packet, evaluator output, runtime observation, or unrun command is evidence or a declaration in its own boundary, not a substitute for the sealed records.

If any comparison fails, write down the exact source ID and field that is missing, stale, conflicting, unregistered, or disallowed, and leave the orientation unresolved. Do not infer a source owner, source contents, defect, repair target, approval, or publication state. The smallest safe next step is to route a permitted follow-up through the owning governed boundary while preserving the original gap. This check supports a traceable operator handoff and nothing more.

Start Here

Audience: onboarding-teammate | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Start Here

This page is the bounded entry path for the Employee Onboarding workstream. It serves future human engineers and AI employees who arrive with no repository context. Its scope is deliberately narrow: a reader can locate approved public-safe source metadata, orient to the product-owned system, and complete one bounded semantic check. This is a documented onboarding route, not a claim that productivity, approval, readiness, or platform authority has been measured. Begin with `content/public/engine-overview.md`, read these sections in order, and keep the three approved source identifiers visible. Before the check, read `docs/employee-onboarding-first-task-contract.md` and `journeys/employee-onboarding-first-task-manifest.json`; the manifest records the journey goals and command allowlist.

Find Authority

Open `sources/authority.json` first. Its `source_files` list identifies the three approved records: `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`; its `employee_may_modify` field is `false`. Then open `registries/sources.json` and match each ID to its public-safe record: the naming decision is a `canonical-platform-source` at `factory:decisions.md#Governed Execution Documentation Steward naming and route`, the documentation boundary is a `product-owned-public-safe-source` at `product:architecture.md#Semantic source`, and the platform delivery contract is a `canonical-platform-source` at `agent-orch:docs/delivery-profile-contract.md#content`. Confirm each record's revision, digest, and `public-safe` disclosure. The external governance references in the authority file are pointers, not authority granted to this page. If a statement is not represented by the sealed IDs and registry records, leave it unresolved rather than adding a source or filling the gap with plausible prose.

Understand the System

Use `architecture.md` for the repository map. `content/` is the semantic source of constrained Markdown; `registries/` records claims, pages, sources, visuals, and protected identity records; and `models/` holds canonical architecture or diagram data. `tools/build_projection.py` deterministically derives `build/index.json` from semantic source, so generated output cannot replace the source. The local release boundary is separate from onboarding: `releases/` and the local `publication/current-release.json` pointer are not touched by this first task, and the repository has no live-directory writer. For orientation, read `models/engine-overview.json` and follow its declared flow from Semantic source to Evidence model to Reading modes. Use that model to understand the product structure, not to infer an unrecorded claim or any external governance decision.

Synchronize a New Content Page

Use this bounded procedure when a future human engineer or AI employee adds a semantic Markdown page.

1. Read `sources/authority.json` and `registries/sources.json`; choose the intended semantic path and page title, and use only source IDs that are registered there with `public-safe` disclosure.
2. Author the page and assign one stable page ID, keeping its metadata block explicit for `visibility`, `audiences`, `reading_modes`, `claims`, and `source_refs`.
3. In the same change, copy the page title, exact path, stable page ID, visibility, audiences, reading modes, claim refs, and source refs into the matching `registries/pages.json` entry. Every claim and source must resolve, and a public page must remain within the public-safe source boundary. This onboarding page is already registered as `page.employee-onboarding-first-task`; preserve that page ID and its existing page-registry entry rather than creating a second page.
4. Run `python3 tools/build_projection.py`, then inspect generated `build/index.json` for the matching page ID, path, title, visibility, claim refs, source refs, content revision, and authority identity. A page is incomplete until its metadata and projection agree.

5. Run the documented deterministic validator, `python3 tools/validate_content.py`, and this route's bounded command, `python3 tools/validate_content.py --journey journey.employee-onboarding-first-task --semantic-only`; retain the observed results as product evidence.

`build/index.json` is generated output, not a hand-edited source. Missing authority or unresolved references stop the procedure; publication, promotion, rollback, and live-directory actions remain out of scope.

Complete Your First Task

From the repository root, first confirm that the contract and pinned journey manifest describe this route, then run the existing bounded content check listed for the journey:

```
python3 tools/validate_content.py --journey journey.employee-onboarding-first-task --semantic-only
```

Retain the exact command and its observed result as product-side first-task evidence. The check exercises the deterministic content validator for this onboarding journey; it does not attest platform authority, employee activation, approval, publication, promotion, scheduling, release-pointer state, or any governing control. If it reports a gap, record the exact gap and keep any repair within the assigned content path. A passing result is evidence about this product documentation route only.

Evidence and Boundaries

Trace onboarding guidance only to `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, as sealed in `sources/authority.json` and registered in `registries/sources.json`. Do not invent product, security, pricing, customer-outcome, or operational claims, and do not copy private governance material or runtime evidence. The separate governing repository owns the employee charter, authority contract, routing mandate, validator authority, activation rules, and publication policy. Product conclusions from this page or its validator remain distinct from those platform-owned controls. Generated projection output is not a source replacement, and this first task does not create a release pointer, promote or publish anything, or write a live directory.

Acceptance Checks

- **AC-EO-1** — Locate `sources/authority.json` and `registries/sources.json`, identify all three authority-listed IDs, and match each to its public-safe registry record, locator, revision, and digest without inventing authority or copying external governance material.
- **AC-EO-2** — Use `architecture.md` to explain the roles of `content/`, `registries/`, and `models/`, distinguish semantic source from deterministic projection, and keep the local release boundary separate from the model flow from Semantic source to Evidence model to Reading modes.
- **AC-EO-3** — From the repository root, read the onboarding contract and journey manifest, complete the exact bounded command in this journey, and retain its product-side result as evidence distinct from platform-owned authority, approval, publication, promotion, scheduling, and release-pointer decisions.

Workstream Queue Selection Receipt

Audience: Lee, downstream auto-orch stage workers, future Internal Knowledge playbook authors, independent readers | Mode: executive, engineer, ai | Provenance: registry-listed (no candidate packet)

Workstream Queue Selection Receipt

Purpose

This pre-authoring, candidate-only receipt gives Lee and downstream auto-orch stage workers a reproducible way to choose the next bounded documentation slice. It makes the comparison inputs, ordered workstream-balance rule, item decision, reader audience, intended outcome, and unresolved authority boundary visible before authoring begins. A reader can therefore reconstruct why the selected item is Internal Knowledge without treating this product evidence as approval, scheduling, publication, promotion, rollback, or platform authority.

Three-Queue Evidence

The comparison contains exactly Internal Knowledge, Employee Onboarding, and Customer Education. The queue records below preserve the current product registry evidence, including age, age rank, eligibility, underserved status, scheduling tier, evidence source and locator, concrete selection or deferral reason, audience, intended outcome, and validation journey. The queue registry does not establish an independent queue-source identity or revision, so that gap remains explicit and no additional provenance is inferred.

Queue ID	Workstream	Queue age / rank	Eligibility	Evidence	Selection or deferral reason	Audience	Intended outcome
`internal`	Internal Knowledge	`3` / `3`	`eligible: true`; `underserved: true`; tier `1`	`artifacts/selection-preflight/repair-brief.json`; `scope.context_workstream and required_selection_rule.rule`	Selected: it is the oldest eligible underserved workstream in the highest eligible tier, before item-level scoring.	Lee and current company operators, including future Internal Knowledge playbook authors and independent readers	Give the mission operator a reproducible, bounded Internal Knowledge selection for the next documentation slice.
`onboarding`	Employee Onboarding	`2` / `2`	`eligible: true`; `underserved: false`; tier `1`	`artifacts/selection-preflight/repair-brief.json`; `scope.comparison_workstreams`	Deferred: it is eligible but not underserved, so the ordered rule excludes it before item-level impact or feasibility.	Lee and current company operators comparing the Employee Onboarding queue	Keep Employee Onboarding visible in the comparison while the mission operator advances the selected Internal Knowledge slice.
`customer`	Customer Education	`1` / `1`	`eligible: true`; `underserved: false`; tier `1`	`artifacts/selection-preflight/repair-brief.json`; `scope.comparison_workstreams`	Deferred: it is eligible but not underserved, so the ordered rule excludes it before item-level impact or feasibility.	Lee and current company operators comparing the Customer Education queue	Keep Customer Education visible as a non-selected comparison queue without widening the selected documentation slice.

The sealed authority inputs remain read-only: `sources/authority.json` has authority revision `2026-08-08`, lists `source.naming-decision`, `source.documentation-boundary`, and `source.platform-delivery-contract`, and sets `employee_may_modify` to `false`; `registries/sources.json` records those three sources as `public-safe`.

Selection Decision

The selected workstream is **Internal Knowledge** (`internal`) for **Lee and downstream auto-orch stage workers**. The ordered workstream-balance rule is: (1) retain the highest eligible scheduling tier; (2) within that tier, select the oldest workstream that is both eligible and underserved; and (3) consider item-level impact and feasibility only after the workstream is selected. All three queues are tier 1. Internal Knowledge is eligible, underserved, and has the oldest age rank, `3`, so it is selected before item-level scoring.

The selected item is **Failure-Triage Decision Guide**, with **impact: 4** and **feasibility: 5**. Employee Onboarding is not selected because it is not underserved at age rank 2; Customer Education is not selected because it is not underserved at age rank 1. Both remain visible, and neither deferral means the workstream is unimportant, infeasible, or permanently excluded.

The success hypothesis is: **If the receipt preserves the exact three workstreams and applies the highest-tier, oldest underserved eligible rule before item-level impact or feasibility, Internal Knowledge and the Failure-Triage Decision Guide remain a reproducible next slice, while every non-selected queue remains visible with its reason.**

Candidate Handoff

The handoff audience is Lee and downstream auto-orch stage workers, with future Internal Knowledge playbook authors and independent readers able to replay the evidence. The intended outcome is a reproducible, bounded Internal Knowledge candidate selection that carries the exact three-queue comparison into the next authoring stage. The validation journey is `journey.replay-selection-decision` in `journeys/workstream-candidate-evidence-selection.json`, supported by `journey.complete-three-queue-evidence` for the row comparison and `journey.confirm-candidate-only-boundary` plus `journey.replay-traced-manifest` for the handoff checks.

This is product evidence only and remains candidate-only and reversible. It does not schedule work, mutate queues, publish, promote, create a release pointer, roll back, or write a live directory. Platform-owned authority and transition attestations remain outside this receipt; the handoff recommends a bounded authoring slice and does not authorize any transition.

Acceptance Checks

- **AC-1 — Complete three-queue evidence:** The reader verifies that the receipt contains exactly Internal Knowledge, Employee Onboarding, and Customer Education, and that every row preserves queue age, eligibility, deferral reason, audience, intended outcome, and validation journey, with the age-rank, underserved, and tier fields needed to replay the comparison.
- **AC-2 — Explicit selected-candidate decision:** The reader applies the ordered rule, confirms Internal Knowledge as the selected workstream, and finds the selected candidate title Failure-Triage Decision Guide with impact 4, feasibility 5, the success hypothesis, and explicit deferral reasons for both non-selected workstreams.
- **AC-3 — Candidate-only boundary:** The reader records that this contract is product evidence only and does not schedule work, mutate queues, publish, promote, create a release pointer, roll back, or write a live directory; platform-owned authority and transition attestations remain unresolved here.
- **AC-4 — Reader replay through a traced journey manifest:** The reader opens the synchronized journey manifest, confirms its non-empty command allowlist, and replays the natural-language journeys whose `traces_to` values cover AC-1 through AC-4 without using an exploratory journey to replace required mission coverage.

These checks make the queue comparison, ordered item decision, candidate-only boundary, and manifest replay reconstructible from stable product paths. They do not authorize scheduling, queue mutation, publication, promotion, release-pointer creation, rollback, or live-directory writing.