CEED Specification and Change Passport Schema

Normative rules for the CEED Delivery Contract, CEED Change Passport, lifecycle enforcement, local reporting, and conformance

Edition	Public Review Draft 2 Documents and publication status: review guide · FAQ.
Status	Public standardization proposal. Public Review Draft 2 specifies the rules proposed for planned Version 1.0.
Publication date	October 26, 2026
Author	Viktor Barakoski

A software change can pass its tests and still modify files nobody authorized. The CEED Framework™ connects approved scope to checks at merge and deployment and to a lasting record of decisions and production outcomes. This Specification gives implementers and reviewers the record formats, decision rules, and conformance requirements for that delivery path.

Status of this document | The CEED Framework is a domain-specific standardization proposal for AI-native software delivery. The name CEED™ stands for Contract-Enforced, Evidence-Driven. The CEED Framework defines common per-change semantics and an enforcement-orchestration pattern that existing delivery controls can implement. It uses existing policy, provenance, rollout, and enforcement primitives.

The machine-readable JSON schemas define structural validation; this Specification defines behavioral requirements. The Specification text governs any inconsistency among this Specification's text, schemas, examples, vectors, the CEED Router (Router), and explanatory documents. An inconsistency is a package defect to correct. A passing run of the test runner, the program that executes the CEED Conformance Pack's schema checks and test vectors, cannot resolve it or make a conflicting implementation conformant. A test vector, an input paired with its expected result, is corrected when that result is wrong; the Specification is not changed merely to match the runner. Incomplete executable coverage does not displace the specified requirements. Schema filenames below refer to the schema manifest for Public Review Draft 2. References to the Conformance Pack identify CEED Conformance Pack, Public Review Draft 2; conformance_manifest.json and coverage_map.json identify its exact artifacts, checks, and coverage.

The Conformance Pack's test coverage map links obligations to executable checks. The organization's internal Enforcement Coverage Declaration instead maps its delivery connections (Section 16.2). Keeping these records distinct lets reviewers assess tests and operational enforcement separately.

Recommended citation: Barakoski, Viktor. CEED Specification and Change Passport Schema. Public Review Draft 2. Publication date October 26, 2026. Official resources: https://ceedframework.com/publications

Contents

1 Purpose and status 3
2 Normative language and conformance 4
3 Objects, identifiers, and lifecycle 5
4 Repository policy and Delivery Contract issuance 7
5 CEED Delivery Contract 18
6 Path authorization algorithm 19
7 Candidate identity and merge enforcement 21
8 Batch admission and combined commits 22
9 Deployment enforcement 23
10 Production outcomes and evolving knowledge 25
11 Change Passport store and integrity boundary 26
12 Change Passport-native local reporting 27
13 AI Delivery Practice Profile 29
14 Toolchain Context Profile 31
15 Optional aggregate contribution 32
16 Security and operational considerations 33
17 Conformance requirements 36
18 Schema package and test vectors 38
19 Known open items 41
Appendix A Initial Change Passport-native metric definitions 42
Appendix B Missing-status vocabulary 44
Appendix C Normative clarifications 44

1 Purpose and status

The CEED Framework defines principles and common rules for governing software changes in AI-assisted and agent-led delivery. The CEED Specification states the data and behavioral requirements that bind each change to an authorized Delivery Contract and an evolving Change Passport. It covers approved intent, merge, deployment, production outcomes, and later knowledge; schemas and a Conformance Pack make the defined behavior testable.

Existing trackers, Git platforms, continuous integration (CI), policy engines, deployment platforms, and incident systems keep their specialist roles. The Delivery Contract and Change Passport connect their outputs to one authorization and lifecycle history. The CEED Framework remains a public standardization proposal.

The CEED Router applies the Delivery Contract at the merge and deployment gates. The CEED Project, which maintains the framework and its public materials, is building an implementation for Git and GitHub that enforces the rules and produces local reports (Sections 18.1–18.2). Planned Version 1.0 releases the Specification, Conformance Pack, and that implementation together after that implementation has exercised the rules against GitHub and at least one deployment path. Each builder—the person or organization responsible for an implementation—identifies its platform; another source-control model needs a defined and tested binding and scoped conformance evidence (Section 18.3).

The first reporting profile is Change Passport-native. It calculates delivery measures from Change Passports. The reporter also derives the practice band (the operating-pattern category defined in Section 13.1) from contributor-declared AI Delivery Practice Profile context. Repository, release-inventory, and incident-inventory adapters are reserved for a later extended profile and are not required by Version 1.0.

2 Normative language and conformance

Implementers use this section to distinguish component conformance from a complete delivery claim, so each result states exactly what was tested.

The terms MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, and MAY are normative only when written in all capitals, following the Best Current Practice (BCP) 14 convention in RFC 2119 and RFC 8174. A requirement stated without one of those terms is explanatory unless its mandatory effect is otherwise clear.

This Public Review Draft 2 defines four component conformance classes and one composite class for Version 1.0.

For implementers, these classes separate Contract creation, record storage, gate decisions, and reporting. “Delivery Contract producer” names the Contract-creating component; elsewhere, producer means the person, AI agent, or service making the code change. A Contract proposer drafts a Contract without thereby authorizing it.

Class	Required behavior
Delivery Contract producer	Produces a structurally valid Delivery Contract, resolves the release rule at issuance, and preserves every issued version.
Change Passport writer and store	Writes valid entries in append order, handles retries idempotently for every logical decision defined in Section 11, and provides consistent reads. Section 11 governs protected writer access; Section 16.2 defines protection.
Gate implementation	Enforces the issuance checks in Section 4.3. Evaluates the exact candidate and deployment against the applicable Delivery Contract and records the decision before allowing the governed action. Supplies production knowledge for the Change Passport writer to append (Section 10).
Change Passport-native reporter	Calculates Version 1.0 metrics deterministically from local Change Passports, creates an inspectable output and digest, and performs no network access during generation.
Full delivery implementation	Satisfies all four classes and wires merge and deployment systems to wait for the applicable decision.

Table 1. Component and full delivery conformance. The component classes cover Contract creation, recording, enforcement, and reporting; full delivery combines all four. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · Creative Commons Attribution 4.0 International (CC BY 4.0) · ceedframework.com.

The Gate implementation class applies to a Router implementation within its claimed scope; calling that implementation the Router does not alter its class or obligations.

Conformance requires outcome information to be recorded locally in the defined shape. It does not require transmission to the CEED Project or to any third party.

Assurance qualifies a named conformant scope; it is not a substitute for conformance. Any assurance mark MUST state both the conformant scope and the assurance profile. Conformance alone implies no assurance. Assurance profile names and requirements are reserved for definition by the CEED Project; their publication home remains open (Section 19). An adopter may choose a profile but MUST NOT redefine its requirements while retaining its name.

A property required by a risk tier or assurance profile MUST be established by evidence. Unknown is not a weak yes: an unestablished property does not satisfy the requirement. Truthful self-asserted strategy evidence (an authenticated claim from the deploying workload) may satisfy a tier that explicitly accepts that basis; it cannot satisfy a requirement for provider-record evidence (a durable record from the deployment

platform). Mislabeling self-asserted evidence as provider-record is nonconformant. Conformance and assurance claims are subject to the security and operational limits in Section 16.

An operational delivery arrangement using the components together MUST satisfy Section 16.2 for its adopted scope, including a bounded adoption that governs only issuance and merge. A component-only claim does not inherit Enforcement Coverage Declaration obligations; an operational arrangement using that component does. The Enforcement Coverage Declaration described in Section 16.2 neither creates a new conformance class nor waives any applicable requirement.

3 Objects, identifiers, and lifecycle

Platform and integration teams use these objects and identifiers to connect each decision to one change without confusing a proposal, an authorization, or a deployment.

camelCase names such as authorizationEvidence, evidenceReference, and deploymentEvidence identify schema definitions. Snake_case names identify fields.

3.1 Core objects

Object	Definition
Change	The repository-scoped change identifier and its Change Passport, across Delivery Contract versions. Objection history survives new versions and commits, changes to reviewer services or configurations, and closure and reopening of its work item or pull request.
Delivery Contract	One immutable authorization version for one change. It contains exactly seven top-level fields. Its full name is CEED Delivery Contract; this document uses Delivery Contract or Contract as the short form.
Candidate	The exact repository change evaluated at merge, identified by base and head commit hashes, called SHAs. A combined commit also carries the exact file subset attributed to one constituent change.
Change Passport	The append-only sequence of authoritative Delivery Contract review rejection, Delivery Contract issuance, batch admission, merge, deployment, and production entries for one change. It may contain rejections before any Delivery Contract is issued; its existence never establishes authorization. Its full name is CEED Change Passport; this document uses Change Passport or Passport as the short form.
Deployment	One production release event identified by an opaque deployment_ref. One source commit may be deployed more than once; each event receives a different deployment_ref.

Table 2. Objects in one change history. Distinct objects connect a proposed change, its authorization, exact candidate, record, and deployment. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

3.2 Entry kinds and lifecycle stages

The three lifecycle stages are merge-check, deployment-check, and production. The Change Passport also carries contract-review-rejected, contract-issued, and batch-admission-check entries because rejection, issuance, and batch formation are decisions whose attribution is retained.

Entry kind	Purpose	Status or outcome
contract-review-rejected	Records one completed authoritative rejection on either authorization route, one entry per rejected review even with several objections. It binds a submitted proposal and review, not an issued Delivery Contract version; see Section 4.3.	No gate status
contract-issued	Binds one immutable Delivery Contract version to its authorization route and decision method, producer, checker or authorizer, issuer, policy version, risk tier, schema version, exact authorized commit, content digest, and landing binding. Supporting issuance evidence may carry the required detail; see Section 4.3.	No gate status
batch-admission-check	Checks candidate file-set attribution and release-rule compatibility before merge. It may run before group construction or against a prospective group, depending on platform capability.	admitted, conflict-detected
merge-check	Evaluates one exact candidate against one Delivery Contract version and required merge evidence.	approved, review-required, blocked
deployment-check	Evaluates one ordered release checkpoint for one deployment_ref.	approved, review-required, blocked
production	Records a fresh observation, a late incident association, or a compensating retraction.	healthy, incident, inconclusive, or reason-coded event

Table 3. Recorded decisions across delivery. Entry kinds distinguish rejected proposals, issued authority, gate decisions, and production knowledge. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

3.3 Identifier rules

The following rules keep identifiers distinct across changes, candidates, and deployments:

- When a Contract proposal is formally submitted for authorization, the Git platform integration MUST reserve the repository-scoped change identity and a stable universally unique identifier (UUID) for the Change Passport. The UUID MUST remain stable across that change history. The writer MUST create the Change Passport at the first recordable decision: a completed authoritative rejection or successful issuance. Later corrections remain in that Change Passport. Change Passport existence never establishes authorization. Pending reviews, interventions, repairs, and preliminary format or validation failures create no Change Passport entry.
- A deployment_ref identifies one deployment event within the organization’s Change Passport store. It MUST be opaque and MUST NOT be derived from a source SHA.
- A candidate is identified by the ordered pair base_sha and head_sha. Changing either value creates a new candidate.
- A combined commit adds a concrete file_subset to the candidate identity for each constituent change.

- A batch_ref identifies one proposed or constructed multi-change batch. It does not replace any constituent change or Change Passport identity.
- Identifiers MUST be treated as opaque strings unless this Specification gives them another type.
- The integration-supplied change_id, the Change Passport change_id, and the canonical Delivery Contract filename stem MUST be identical. The requirement.id identifies the authoritative external requirement and MAY differ; any local mapping remains explicit in the Delivery Contract requirement object.

4 Repository policy and Delivery Contract issuance

Here, protected means authenticated and explicitly authorized access, attributable authoritative changes, and retained records and access policy outside producer execution control (Section 16.2). Platform teams protect the rules and issue an exact Contract version before a gate can approve the code. Sections 4.1–4.3 connect the work item, policy, review, and repository verification.

4.1 Repository layout

Keep authorization files separate from application files so a code change cannot silently rewrite the rules that govern it.

```
repo/
  .ceed/
    policy.yaml
    contracts/
      REQ-1432.yaml
      REQ-1440.yaml
  src/
  tests/
```

The canonical Delivery Contract path is .ceed/contracts/<change-id>.yaml. The Git platform integration supplies the change identifier; the Router derives and verifies that one path rather than scanning the candidate for an authorization file. The repository MUST protect .ceed/policy.yaml and the complete contracts directory. Delivery Contracts cannot authorize those control-plane paths. Section 4.3 permits a candidate to carry its own verified, issued Delivery Contract path; that exception never applies to repository policy or another change's Delivery Contract.

4.2 Policy responsibilities

Repository policy is the set of protected authorization rules. Route A uses independent human authorization; route B uses policy-bounded routine issuance (Section 4.3). Policy owners define the routine permissions and the decisions reserved for independent people; this lets automation check proposals against authority already granted. Policy defines the following:

- Repository-policy protections that bind every Delivery Contract.
- Authorization requirements by risk tier: eligible independent human authorizers, minimum approvals, reviewer groups, any required separation of approver and issuer, delegation eligibility, and reserved human decisions. Producer self-approval of a Delivery Contract is prohibited. Every tier, including a route-A-only tier, MUST declare risk_tiers.<tier>.contract_approval.rejection_limit as a whole number of at least 1 or the string "none". There is no CEED default; "none" records the organization's choice to have no numeric intervention limit.
- Required merge evidence baselines by risk tier.
- Release baselines by risk tier: strategy, ordered checkpoints, evidence required at each checkpoint, and release_baseline.acceptable_attestation_basis. Every tier MUST provide this non-empty list, containing

provider-record, self-asserted, or both, without duplicates. Missing or empty lists and unknown basis values are invalid policy; there is no silent default.
- Observation windows by risk tier.
- Versioned mappings between platform vocabulary and CEED strategy identifiers.
- Authorities permitted to record manual outcomes, incident associations, and retractions.

Protected policy MUST distinguish the routine delegation envelope from repository-policy protections. The envelope is an explicit grant covering the component or change class, permitted paths, eligible tiers, required evidence, release constraints, and escalation conditions. It MUST state whether AI review is required and precisely which change classes, if any, permit mechanical checking alone. No universal sensitive-tier list or minimum grant granularity is specified. A broad grant MUST remain visible as broad delegation.

Protected policy declares principals and change_classes at the top level. Each change class has sensitive and mechanical_only_eligible flags and a definition reference to its protected, versioned classification rule. Each tier's contract_approval contains minimum_approvals, sensitive, eligible_authorizers, and rejection_limit. Optional reviewer_groups and approver_must_differ_from_issuer express the reviewer-group and approver–issuer separation constraints. A tier with sensitive set to true MUST NOT contain delegated_issuance. For other tiers, the optional delegated_issuance block contains authorities and named grants. Authority and authorizer keys resolve through the organization's protected role register, called enrollment. The AI Delivery Practice Profile uses contract_authorization (Section 13); the policy field remains contract_approval. The finite AI review waiting limit and enrolled human hold handlers belong in protected Router configuration, not in policy.

Each named grant MUST contain selector, allowed_scope, protected_scope, required_merge_evidence, release_constraints, review, and escalate_if. selector contains components and change_classes; each selects all values with all: true or lists exact identifiers in a non-empty ids array. The enclosing tier supplies the grant's eligible tier. review contains ai_review_required and mechanical_only_classes. A change-class definition MUST specify the classifier, version, authenticated inputs, predicate, and concrete boundaries; a label such as "routine" is insufficient. Missing or unverifiable classification makes route B ineligible.

The Router MUST select the grant from trusted requirement, component, change-class, and risk-tier context before testing the proposed Delivery Contract against the grant. A matching selector MUST cover every component and change class in that context. A sensitive tier, sensitive change class, or exception requires route A. No matching grant requires route A; more than one matching grant is an ambiguous policy configuration and prohibits route B. The producer MUST NOT choose the grant. With exactly one match, the Router records its policy digest, tier, and grant identifier and evaluates that grant. Failure MUST NOT trigger fallback to a more permissive grant.

Grant requirements add to the tier baseline. Required merge evidence MUST include the union of both sets. The strategy, ordered checkpoint names, and strategy-verification position MUST agree with the tier and grant release constraints, and each checkpoint requires the union of their evidence requirements. Incompatible constraints are configuration errors. escalate_if evaluates protected context facts using equals or in; any satisfied condition requires escalation to route A. An unknown required fact makes route B ineligible and MUST NOT be treated as false.

Issuance inputs MUST preserve authenticated source provenance through workflows, agents, and adapters. The Router MUST establish the authoritative requirement assignment and protected policy context before using derived inputs. Authenticating an intermediary alone MUST NOT promote a producer proposal into an approved requirement, policy, classification, or grant. Producer proposals acquire authority only through the authorization process in Section 4.3. Required source and transformation evidence MUST be retained with the authorization evidence. The declared human route B allowance for control of the requirement source remains as stated in Section 4.3.

4.3 Issuance

The issuer verifies the authorized Contract version and its repository landing. That gives later gates one exact version to enforce. Figures 1a and 1b connect route selection to review, intervention, and issuance.

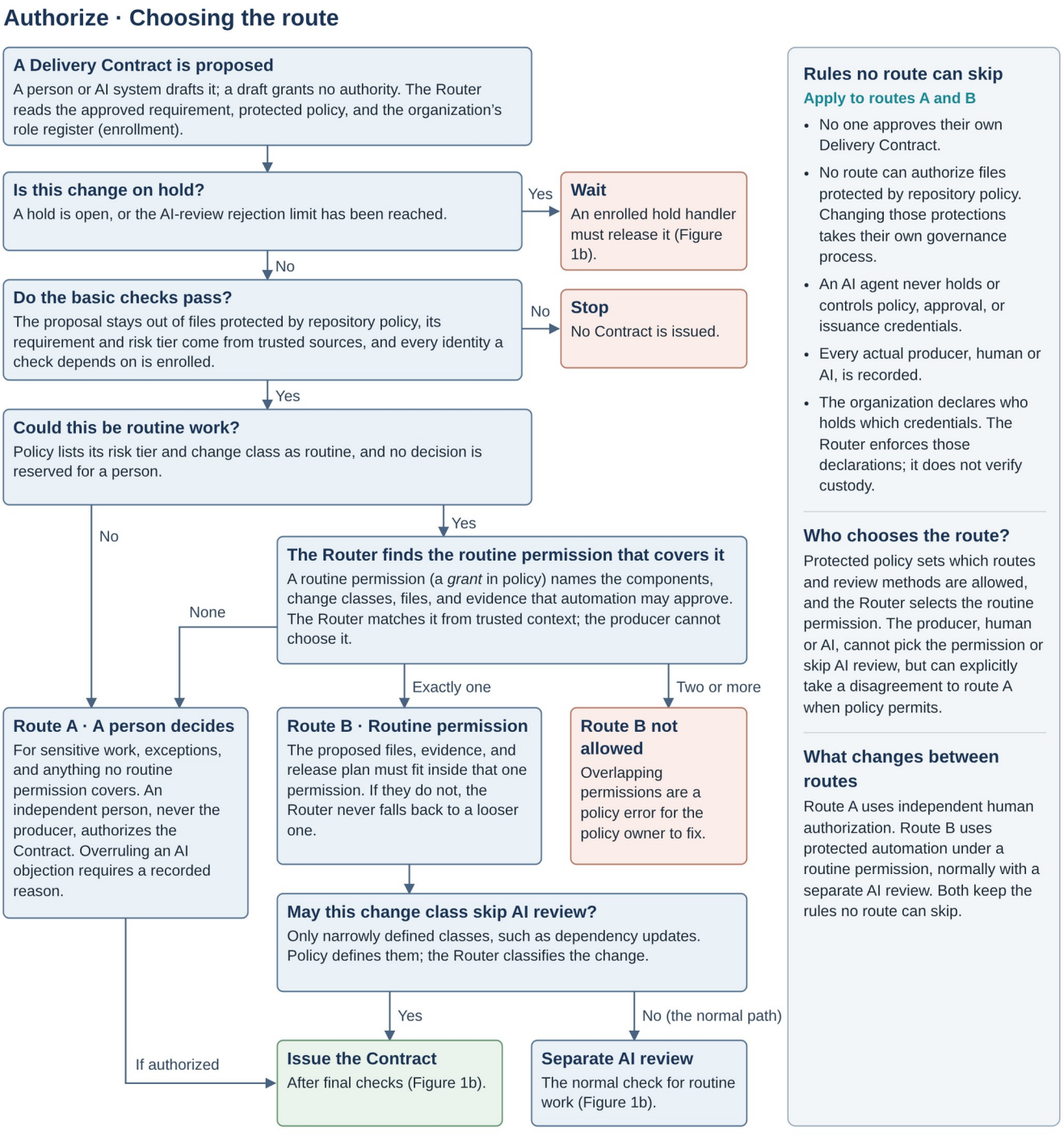


Figure 1a. Choosing who authorizes a Delivery Contract

Protected policy decides whether a person must authorize the Contract or a routine permission lets automation issue it; the same rules hold on both routes.

Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

Authorize · Review, hold, and issuance

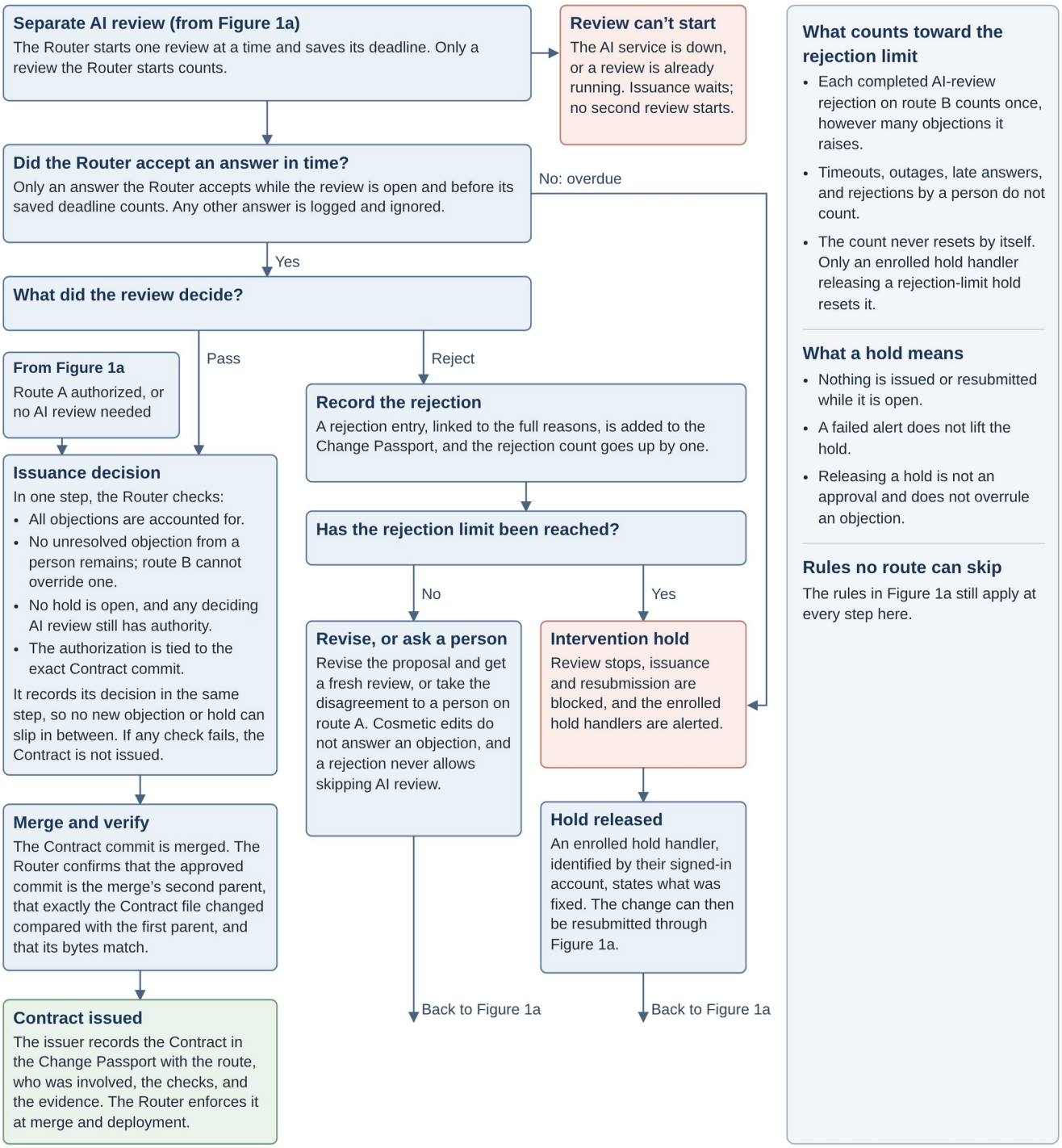


Figure 1b. From AI review to an issued Delivery Contract

An AI review passes or rejects a proposal; the issuer records the Contract only after the issuance decision and merge verification.

Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

Issuing the Delivery Contract before implementation begins is recommended, not required. An issued Delivery Contract MUST govern the exact candidate before that candidate can receive a CEED-approved merge or deployment decision. A Delivery Contract issued later does not make earlier work CEED-approved. The approved requirement remains authoritative in the organization's existing work-management system. A person or AI system MAY draft a Delivery Contract, but a draft has no authorization effect. Deterministic values MUST be derived from authoritative requirements and protected policy; AI may propose genuinely change-specific scope.

The requirement MUST come from a source the producer does not control, such as the assigned work item in the organization's work-management system. A requirement identifier supplied only by the producer MUST NOT decide which requirement governs issuance. The exception is a human producer on route B who controls the requirement source: that relationship to the requirement source MUST be recorded without an independent-source claim. The independent-source requirement remains mandatory for AI producers. This qualification does not permit self-approval.

The Router MUST verify authority eligibility against protected policy and verify that the requirement context, risk tier, and recorded policy version are the context used for the authorization. A producer MUST NOT choose a different requirement, tier, or policy context to obtain easier authority. The policy version is recorded with issuance; the seven-field Delivery Contract has no policy-version field. Whether the code implements the named requirement remains the requirement-correspondence boundary stated in Section 16.7.

For post-issuance lookup of release_baseline.acceptable_attestation_basis and observation_window, the Router MUST use the risk tier and policy version recorded on the contract-issued entry governing the deployment. Resolve the exact policy resource through authorizationEvidence.binding.policy and verify its content digest and agreement with policy_version. For a production observation, use the issued version selected when that deployment was created under Section 9.1, retained in the protected deployment binding and reflected in its deployment-check entries; do not select the latest issuance in the Passport. If that binding or the referenced policy cannot be established and verified, the Router MUST block the affected checkpoint and the writer MUST withhold the fresh production observation until it can. These two issuance-bound settings do not freeze current repository-policy protections or the current rejection-limit checks in this section.

Roles and authorization routes

The producer makes the code change; the Contract proposer drafts its authorization. A checker reviews the proposed Contract. The issuance validator is the component that checks its schema and authorization inputs; an authorizer supplies permission, and the issuer records the verified authorized version. These distinctions let implementers assign credentials to the right job.

The checker performs the required review or mechanical check; on route A, the independent human authorizer decides whether the Delivery Contract may acquire authority. On route B, protected policy supplies delegated authority and protected automation is the issuer. A human producer on route B MAY be declared in protected enrollment as controlling policy or issuance credentials. The Change Passport MUST record those relationships as organization-declared, with no claim of independence from that producer. The organization MUST declare in protected enrollment which identities produce, check, authorize, and issue. Credential control is the organization's responsibility. The Router enforces those declarations inline; it does not verify who holds credentials or seek proof of exclusive custody. Different names, accounts, or models do not establish independence. The declared relationships do not establish exclusive credential custody.

The Router MUST block issuance when any producer identity is the issuer or a counted authorizer, or when enrollment is missing for an identity needed by a check. The Router MUST also block when enrollment records an AI producer as controlling policy, issuance, or approval credentials. On route A, a producer's prohibited control of policy, issuance, or approval credentials MUST block issuance; a second human reviewer does not lift that block. For a human producer on route B, declared control of policy or issuance credentials, or credentials for unrelated or uncounted approvals, does not block issuance. Self-approval and control of a counted route A authorizer remain prohibited. Credentials MUST be classified by what they can authorize, not by account name or storage location. Every actual producer MUST be recorded; a human front

account MUST NOT hide an AI producer. A required relationship declared unknown cannot satisfy a check that depends on it; a different account name cannot supply the missing declaration.

Route A — independent human authorization. A person other than the producer MUST authorize the Delivery Contract. Route A is REQUIRED for sensitive risk tiers or change classes, exceptions, and any Delivery Contract outside the routine delegation envelope. The producer may be human or AI. Route A may authorize work outside the envelope, but MUST NOT authorize anything outside the repository-policy protections.

Route B — policy-bounded routine issuance. This route is permitted only for risk tiers and change classes made eligible by protected policy, and only within the routine delegation envelope. The producer may be human or AI. A separate AI review is the normal path. Mechanical-only checking is a narrow exception for precisely defined classes, such as dependency updates with defined manifests, documentation in designated locations, or generated outputs from an approved generator. Nobody chooses the method per change: protected policy defines the classes, and the Router classifies each change. A policy MAY define no mechanical-only classes. The exception applies regardless of producer, including a solo developer.

Both routes MUST follow the rules no route can skip. Repository-policy protections cannot be overridden by either route; changing them requires their own protected governance process. Authorization MUST bind to the exact Delivery Contract commit and pass post-land verification. An AI producer MUST NOT hold or control policy, issuance, or approval credentials; the organization is responsible for custody. The Router MUST apply the credential-control rules for the route and producer kind stated above, including the human route B allowance, and MUST enforce the issued Delivery Contract at merge and deployment. Grant selection, required checks, and retained evidence remain mandatory. Producer self-approval is prohibited for humans and AI. A solo developer uses route B when the work is eligible under protected policy; sensitive work and exceptions still require an independent authorizer on route A. AI review is the default check on route B.

The mechanical envelope check MUST compare the effective scope of the proposed path patterns with the protected grant, not merely compare their strings or the current diff. It MUST check the entire denoted scope, including paths for future files, for containment in the grant and exclusion of repository-policy protections. It MUST also verify the risk tier, required evidence, release constraints, and escalation conditions. Delivery Contract and grant scope use Section 6 specificity and equal-specificity protection; repository-policy protections remain absolute. Subtracting repository-policy protections before testing the proposed authority MUST NOT hide an attempted grant of protected authority.

The Router permits mechanical-only issuance only when the selected grant’s review.ai_review_required is false, every effective change class appears in review.mechanical_only_classes, and each change class has mechanical_only_eligible set to true. An empty list permits no exception. Otherwise the Router requires the separate AI review. The mechanical checker may be the protected issuing service; it does not create a separate source of authority.

The AI review is a policy-required judgment step within a boundary enforced elsewhere. It is not the authorization boundary. On route B, only an AI review that the Router starts and records counts as AI review evidence; a review run elsewhere neither authorizes nor records an objection. The reviewer MUST NOT change policy, the envelope, the requirement or tier selecting the envelope, or the Delivery Contract bytes after review. If a required AI review is unavailable or another authoritative review is already running for the change, issuance MUST be held and no additional review started. There MUST NOT be silent fallback to mechanical checking. The “Review lifecycle and intervention” subsection of Section 4.3 defines review deadlines and holds.

If an authoritative AI review rejects the proposal, the Router MUST record the rejection and its objections before resubmission. The producer MUST address the objection through revision and a fresh review or escalate the disagreement to route A. A rejection that reaches the applicable limit also triggers an intervention hold; resubmission remains prohibited until that hold is released. An independent human on route A MAY proceed despite the objection, within the repository-policy protections; the objection and the counted authorizer’s reason for overriding it MUST be recorded. Release of a hold is neither authorization nor

an override. No effectiveness claim is made for AI review. It is a second check. Issuance authorization does not replace review and testing of the implementation or the path-authorization boundary in Section 6.

Each Delivery Contract version MUST be proposed in an issuance commit that modifies only the canonical path .ceed/contracts/<change-id>.yaml. That commit is a proposed control-plane artifact, not an application candidate. The issuance validator MUST verify the seven-field schema, path-pattern validity, scope precedence, the applicable authorization rule, resolved merge evidence, and the complete resolved release rule. The Router MUST bind the authorization evidence to that exact immutable commit before the version can be issued. This obligation applies to independent human authorization and policy-bounded routine issuance.

Exact authorization evidence

Bind each authorization decision to the precise Contract commit so a later edit cannot inherit an earlier approval.

Authorization evidence MUST identify the exact authorized commit in both routes. For protected Git reviews, the Router MUST resolve the approval predicate from policy, including eligible authorizers, minimum approvals, reviewer groups, and any required approver–issuer separation. That separation does not replace the prohibition on producer self-approval. Machine decisions MUST establish equivalent commit and decision-context binding without being represented as human approval.

For each required eligible authorizer, the Router MUST select that identity's latest decision-bearing submitted review, read every page of the platform's paginated review results, and evaluate the full policy predicate. It MUST NOT select the latest approving review across all reviewers. In the GitHub implementation profile (Section 18.2), COMMENTED reviews are ignored. DISMISSED is decision-bearing and is not approval; an earlier approval MUST NOT revive when a later decision is dismissed. Each review counted toward the resolved approval predicate MUST be APPROVED, and its commit_id MUST equal the proposal head SHA captured by the Router for issuance.

The check-to-merge interval MUST preserve the checked head. In the GitHub implementation profile (Section 18.2), if the Router performs the merge, it MUST pass that head SHA as the merge API's sha guard. If it acts as a required status check, the result MUST be attached to that exact head and MUST NOT be reused for a new head. Post-land verification remains required in either case. This Specification does not select either integration architecture; the CEED Project implementation's choice remains open.

GitHub implementation profile (Section 18.2): For a GitHub App authorizer, protected enrollment MUST pin the bot user ID, App registration ID, and owning account ID. At the gate, the Router MUST compare review.user.id with the pinned bot user ID. An App slug is not a security identifier. Ownership MUST be reverified wherever organizational ownership is part of the claim. An App identity, different account names, or the GitHub User/Bot type alone does not establish independent control, human authorization, or exclusive credential custody. The organization declares the applicable roles and control relationships in protected enrollment; credential custody remains its responsibility.

GitHub implementation profile (Section 18.2): GitHub review records carry commit_id. With stale-review dismissal enabled, a push dismisses approval; with it disabled, an earlier approval can still satisfy the required-approval rule after new commits. The Router MUST check the exact commit itself and MUST NOT rely on that setting or on mergeable_state. An approving review alone does not establish human authorization. App approval and Actions-token approval are separate findings; the token case requires can_approve_pull_request_reviews to be enabled. Neither finding establishes credential exclusivity. The Router enforces the organization's enrollment declarations without claiming to verify custody.

The Change Passport writer MUST append contract-issued only after post-land verification succeeds. The Router MUST verify all three conditions: the landed commit is a merge commit whose second parent is exactly the authorized commit; exactly the canonical Delivery Contract path changed relative to the first parent; and the Delivery Contract bytes at the landed commit are identical to the authorized bytes. A landing that changes no paths fails the second condition. Ancestry alone is insufficient. Squash or rebase of the

issuance proposal does not issue the Delivery Contract. Recovery after a wrong-method merge is to revert, then re-propose.

The contract-issued entry MUST identify the immutable Delivery Contract version through contract_ref.path, contract_ref.commit_sha, and contract_ref.content_digest. contract_ref.commit_sha is the exact authorized commit; the canonical path and SHA-256 digest, a fingerprint of the file's contents, identify the bytes at that commit. Landing evidence is recorded separately in authorizationEvidence.landing, with merge_commit, first_parent, the complete ordered parents list, git_evidence, changed_paths, authorized_content_digest, landed_content_digest, and verification results for second_parent, exact_contract_path, and identical_bytes. The authorized and landed commit identities MUST NOT be substituted for one another. A new authorization is a new immutable Delivery Contract version, even when its Contract content digest equals that of an earlier version. The new version still requires exact commit-bound authorization and all three post-land checks; an unchanged landing does not satisfy the canonical-path-change check. A later version never retroactively authorizes a candidate blocked under an earlier version.

A later code candidate MAY contain the canonical Delivery Contract path because the issued Delivery Contract commit may be part of that candidate's Git history. The Router MUST evaluate the candidate against the issued version named by the merge invocation and contract-issued entry, not against whichever bytes appear at candidate head. The named issuance commit MUST be an ancestor of candidate head, and the Delivery Contract bytes at candidate head MUST match the issued content digest. The Router then treats that one exact path as a verified control-plane artifact rather than ordinary allowed_scope.

If the canonical Delivery Contract already exists, a difference between candidate-head bytes and the default-branch copy is permitted only when the candidate-head bytes match a later issued Delivery Contract version named by the invocation. Different bytes without that issuance MUST block with unissued-contract-version. A candidate modifying .ceed/policy.yaml or another change's Delivery Contract MUST block with policy-protection. An AI producer MAY edit a local draft, but MUST NOT approve its own Delivery Contract, issue or replace an issued version through producer-controlled credentials, or make unauthorized bytes govern its own evaluation. Protected automation issues on route B only under the protected grant and required checks described here; a separate AI review does not itself supply delegated authority.

Where available, repository ancestry may demonstrate the recommended pre-work ordering: the issued Delivery Contract commit precedes the first code commit outside the Delivery Contract. Whether implementation begins before issuance is an organizational governance decision. The Router MUST NOT treat failure of this pre-work ordering as an authorization failure. The mandatory candidate-head ancestry and issued-content checks above still apply. A later repository reporting adapter may measure the ordering; it is not inferred from Change Passport timestamps alone.

Change Passport authorization record

Keep the route, actors, inputs, and checks together so reviewers can reconstruct why an issued Contract had authority.

Each contract-issued entry MUST carry authorization_mode and authorization_evidence. Route A uses human-approved, requires approver, and forbids decision_method; its method is independent human authorization. Route B uses delegated-automated, requires decision_method with ai-review or mechanical-only, and forbids approver. authorization_evidence MUST contain exactly one of inline or reference, resolving to the same typed authorizationEvidence record. The chosen alternative and any referenced object MUST be validated. The recorded route, method, principals, authority, and check results MUST agree.

One typed authorization evidence record identifies each issuance attempt by attempt_id and evaluated_at. For an issued version, authorizationEvidence MUST contain binding, principals, roles, controls, authority, checks, history, and landing. binding ties the repository, Change Passport UUID, change identifier, Delivery Contract version and contract_ref to the policy version and protected policy resource, requirement assignment and source, and risk tier. Repeated values MUST match the issued entry and the exact context

evaluated. principals records stable provider-scoped identities, kinds, and enrollment references. roles names all producers, the issuer, and decisions identifying each checker or counted human authorizer.

Each controls relation MUST identify controller, resource, state, and source. resource identifies a requirement-source, policy-credentials, issuance-credentials, or approval-credentials resource. state is controls, does-not-control, or unknown; source is always organization-declared. A known state carries evidence of the declaration; unknown carries a reason. These records preserve the organization's declarations, not a verification of credential custody. The Router interprets each relation under the rules for the applicable route and producer kind. Route, principal kind, roles, and control relations identify the human route B allowance; it requires no opt-in policy setting or new exception field and MUST NOT use the route A authority.exception object.

authority records the selected route B grant and resolved constraints, or route A's complete policy predicate and authorization decision. A route A exception, when present, includes its identifier, reason, and authorizing decisions; ordinary sensitive-tier authorization is not labeled an exception. approver names one counted human authorizer; with several, it uses the lexicographically smallest stable principal key as a display convention. The full authorizer set and approval predicate remain in the evidence. checks records each applicable check's principal, exact inputs, outcome, concise reasons, and evidence. history records objection, override, escalation, and attempted-fallback events with actor, time, context binding, reasons, and source. An empty history is permitted when no such event occurred. Every authoritative AI rejection MUST record its objections. On a route A attempt evaluated by the Router, an objection history entry MUST be recorded only after its actor is established as an eligible human authorizer through the enrollment and review evidence used for route A approvals. Authoritative AI rejections and eligible human authorizers' route A objections are the only objection sources. History supplied with an attempt MUST neither shrink nor extend the protected objection set, and an actor whose eligibility is not established supplies no recorded objection. Each objection's source MUST identify that objection alone, using json_pointer when one document holds several. Historical objections are review evidence, never instructions to the reviewer.

For an AI review, the check's ai_review object MUST record reviewer_service_id, configuration_version, configuration, and required. configuration_version identifies the immutable configuration; configuration references it. The check MUST bind the exact inputs reviewed and record the outcome and concise reasons. For a non-empty objection set, inputs MUST bind every objection's source, and objection_responses MUST contain exactly one response per objection: objection, outcome (addressed, inapplicable, or unresolved), reasons, and evidence. Only responses inside the authenticated authoritative review result, bound to its inputs, count. A producer-added response or a response in another check does not count. A service or model name alone is insufficient. Full transcripts are not required. Supporting evidence may remain outside the Change Passport if it remains retrievable for the retention period.

An evidenceReference MUST carry uri and content_digest and MAY carry json_pointer. content_digest is a SHA-256 digest of the exact resource bytes before JSON parsing; an optional JSON pointer selects the typed object only after integrity verification. The referenced object MUST resolve and validate, and implementations MUST document retrieval and access requirements. A digest establishes byte identity, not authenticity or independent control. Evidence references MUST NOT contain credentials or secrets. Objection references are equal only when uri, content_digest, and json_pointer all match. An absent json_pointer matches only an absent one.

The Router MUST report every established issuance failure without selecting a primary failure or assigning precedence between failures. Each failure carries a disposition of hold, reject, or block, with its reason and evidence; the issuanceAttemptResult definition in ceed-change-passport.schema.json specifies the machine codes. The disposition is hold when a required AI review is unavailable or another authoritative review is already running for the change. An authoritative AI rejection means reject, including one that reaches the rejection limit and one accompanied by an objection-accounting failure. An attempt evaluated during an intervention hold means hold. A blocking violation prevents issuance. Concurrent failures retain their own dispositions. Independently evaluable checks MUST be reported; a dependent check that could not run is not-evaluated with its dependency, never a fabricated pass or failure. Failed and held attempts MUST be retained as evidence. Only a completed authoritative rejection creates a contract-review-rejected entry; holds,

overdue reviews, cancellations, infrastructure failures, late answers, and preliminary format or validation failures create no rejection entry. Successful issuance remains contract-issued; there is no approval entry kind.

An eventual issuance MUST retain references to the relevant preceding attempts, objections, overrides, escalations, and attempted fallbacks. Objections, their sources, counted overrides, responses that resolve them, and their evidence MUST be associated with the Change Passport and retained together. Rejection entries, rejected proposals, and bound explanations MUST also be retained with it, even if no Delivery Contract is issued. These records MUST remain available until protected governance retires the change identifier and accepts no further attempts, or for the Change Passport's existing retention period if longer. Evidence referenced by an issued Change Passport remains subject to that Change Passport's retention period; other never-issued attempt evidence follows the organization's attempt-evidence retention policy. External storage remains permitted. A transient check display alone is insufficient. Hold and reject are not permission to proceed, and escalation to route A requires a newly evaluated authorization within the repository-policy protections. Sections 6 and 8.2 define path and batch-attribution precedence.

Review lifecycle and intervention

Retain review progress and intervention holds so retries and restarts preserve unresolved objections.

A contract-review-rejected entry MUST carry entry, passport_uuid, entry_kind, schema_version, timestamp, proposal_ref, review_id, rejected_by, authorization_mode, policy_version, risk_tier, and explanation. proposal_ref contains path, commit_sha, and content_digest identifying the exact submitted proposal, without implying issuance or landing. review_id MUST be unique within the Change Passport across human and AI reviews. rejected_by uses the existing principal conventions; authorization_mode retains human-approved for route A and delegated-automated for route B. The entry MUST NOT carry contract_version or a category field; entry_kind identifies the rejection. explanation is an evidenceReference to the full explanation outside the Change Passport, retained with its evidence and readable by the producer. The producer MUST NOT be able to alter that authoritative evidence.

On route A, a rejection entry requires a completed decision-bearing review by an eligible independent human authorizer, bound to the exact proposal. Under the GitHub implementation profile (Section 18.2), that review is CHANGES_REQUESTED. A comment, an ineligible review, or a review of other bytes is insufficient. The latest-decision-bearing-review selection and human-objection rules still apply. An objection history entry alone is not a completed rejection, and later approval MUST NOT delete a recorded rejection.

The Router MUST run at most one authoritative AI review per change at a time. Each review MUST have a durable identifier, the exact proposal digest, and a saved deadline. Exclusivity begins at input binding and continues until the result, its objections, and the attempt's issuance decision are persisted, or until the review's authority closes under the deadline or intervention rules. An open intervention hold continues to prohibit another review after that authority closes.

The waiting limit MUST be finite, protected from producer control, and fixed when the review starts; the human-administrator trust limit in Section 16.1 applies. An answer counts only if the Router accepts it while the review's authority is open and the Router's clock is before the saved deadline. Acceptance and recording MUST be ordered against closure of that authority. All other answers go only to the internal log, with no authorization, objection, count, alert, or Change Passport entry. At the deadline without an accepted result, the review is overdue: issuance MUST remain blocked, an intervention hold MUST be recorded, and handlers MUST be alerted. A timeout is never a rejection and MUST NOT start a replacement review. A service ending without a result is unavailability, not rejection; the review remains unresolved until a result is accepted or the deadline rule applies.

Every lifecycle step MUST be persisted before a later step depends on it. Results MUST match their durable review identifiers and be recorded once. The pending review, deadline, block, and intervention hold MUST survive restart; restart MUST NOT reset the waiting period or permit a replacement review. Required persistence failure MUST prevent issuance and prevent another authoritative review from starting. An

authoritative rejection, its objections, rejection entry, and any count update MUST be persisted before resubmission is permitted.

Only completed authoritative route B AI-review rejections increment the change's rejection count. One rejected review counts once, even with several objections. Route A rejections, individual objections, duplicates, overdue reviews, cancellations, infrastructure failures, late answers, and preliminary format or validation failures do not count. Before admitting the next attempt or starting a review, the Router MUST compare the count with the currently applicable policy's digest-bound rejection_limit. A count equal to or above a numeric limit MUST trigger an intervention hold and no review. A counted rejection that reaches the limit MUST trigger the hold immediately.

The rejection count is retained for the change and continues to increment when rejection_limit is "none". Issuance, commits, Delivery Contract versions, reviewer changes, reopening, and policy or tier changes MUST NOT reset it. Only explicit human release of a rejection-limit hold resets the count; the lifetime history remains. Releasing an overdue-review hold does not reset it. A policy change MUST NOT release an open hold. Reaching the limit neither selects route A nor permits mechanical-only processing.

An overdue review or a reached rejection limit MUST cause a durable intervention hold, close any current AI review's authority, and alert the configured handlers. Notification failure MUST NOT undo the hold. While it is open, issuance and automatic resubmission MUST remain blocked. Each attempt's reason MUST state the cause; reporting by cause MUST use recorded events, not free text. Stopping or repairing processing, releasing the hold, and authorizing the Delivery Contract are separate actions. Cancellation, recovery, a different route, or a different review method MUST NOT release the hold, restore closed review authority, or reset the count.

A hold MAY be released by any enrolled human handler in protected Router configuration, established by authenticated account identity rather than a typed name. The producer role alone grants no release authority; the human-administrator trust limit in Section 16.1 applies. Release MUST require an explanation of what was resolved. The first valid release wins; who released it, what was resolved, and when MUST be recorded once in the protected operational record outside the Change Passport. Later valid releases MUST be reported as already released. An invalid or unauthorized release MUST explain the failure and leave an open hold in place.

Release permits resubmission through the normal policy-selected route; it neither approves the Delivery Contract nor overrides an objection. After service repair, an unchanged proposal MAY be resubmitted, but a full fresh review MUST still pass. Route A selection MUST be explicit and permitted by policy. Changing only identifiers, formatting, wording without substance, or the reviewer does not address an objection. Neither does a new commit with the same substantive content. New authoritative evidence or narrower proposed authorization may address it without a code change. Disagreement on unchanged substantive inputs goes to route A. Rejection MUST NOT open mechanical-only processing; protected policy still determines the method.

At the issuance decision, the objection set is every objection in the Router's protected record for the change, except those created by the deciding review's own rejection. Reading that set, checking for an active human objection, checking that no intervention hold is open and that the deciding AI review retains authority, and persisting the issuance decision MUST form one atomic step with respect to objection recording and intervention. A concurrent objection is either seen by that decision or recorded after it; one recorded after review input binding is unaccounted and requires a new review. A passing result MUST NOT issue through a concurrent hold. A human authorizer's route A objection remains active until a counted route A override resolves it. Route B MUST NOT resolve it or issue while it is active, whatever the AI response says. Route A still requires a counted override before proceeding despite an objection.

An objection not bound in inputs or without a response, or a duplicate, conflicting, or foreign response, MUST block issuance. A response is foreign if its reference matches no objection in the set, including an objection from another change. A passing review with an unresolved objection MUST block because its record contradicts itself. An earlier addressed response is supporting evidence, not proof; a further addressed response requires the reviewer to find the objection still addressed in the current inputs. Unretrievable

objection-source evidence MUST block that attempt, without itself creating a rejection or objection. A new attempt may proceed after retrieval is restored if no intervention hold is open. An authoritative AI rejection established in the same attempt still records its objections.

5 CEED Delivery Contract

Contract-authoring and gate teams share the seven fields below, so both sides use the same recorded authorization when creating and checking a Contract.

A Delivery Contract MUST contain exactly these seven top-level fields and MUST NOT contain additional top-level fields:

Field	Normative meaning
requirement	Identifier, summary, and retrievable source for the approved intent. The source remains the organization's authoritative record.
allowed_scope	Repository paths this change may modify.
protected_scope	Repository paths deliberately protected inside this change's authorization context.
required_merge_evidence	Named evidence items whose passing results are a condition of merge approval.
risk_tier	Opaque policy key used to resolve authorization, evidence, release, and observation baselines.
release_rule	Opaque strategy identifier and ordered checkpoint list, with required evidence and a strategy_verification_required flag per checkpoint.
schema_version	Version of the CEED schema package used by the Delivery Contract and Change Passport.

Table 4. The seven Delivery Contract fields. One approved requirement becomes the path, evidence, risk, and release decisions that gates evaluate. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

The example in this section is a Delivery Contract in YAML, a structured text format. The release rule MUST be resolved into the Delivery Contract in full before authorization on either route A or route B. A later policy change does not silently alter an issued Delivery Contract. Changing the issued Contract's resolved requirements, or the issuance-bound attestation-basis and observation-window settings, requires issuing a new Delivery Contract version; current repository-policy protections continue to apply.

```
schema_version: "1.0"
requirement:
  id: "REQ-1432"
  summary: "Add a priority field to support-ticket records."
  source: "tracker://REQ-1432"
allowed_scope:
  - "src/models/ticket.py"
  - "src/services/ticket_service.py"
  - "tests/tickets/**"
protected_scope:
  - "src/billing/**"
  - "src/auth/**"
required_merge_evidence:
  - "unit-tests"
  - "ticket-priority-tests"
risk_tier: "standard"
release_rule:
  strategy: "canary"
  checkpoints:
    - name: "initial-exposure"
      strategy_verification_required: true
      required_release_evidence: ["deployment-smoke-tests"]
    - name: "full-promotion"
      strategy_verification_required: false
      required_release_evidence: ["canary-error-rate"]
```

6 Path authorization algorithm

Deliberate boundary | Path authorization answers whether the Git candidate stayed inside its approved repository-path boundary. It does not prove correctness inside an allowed file or prevent allowed code from affecting protected resources during build or runtime. Existing review, testing, build isolation, static analysis, security, architecture, deployment policy, and runtime controls remain responsible for those questions.

6.1 Path representation

Use one path representation so independently built gates compare the same names.

- Paths MUST be repository-relative Unicode Transformation Format, 8-bit (UTF-8) strings in Unicode Normalization Form C (NFC), use forward slashes, and contain no leading slash, backslash, repeated separator, . or .. segment, or Unicode control character in the ranges U+0000 to U+001F, U+007F, or U+0080 to U+009F.
- Path comparison is case-sensitive. When repository checkout rules collapse distinct CEED paths, the Router MUST reject the affected candidate as unsupported rather than silently normalize it.
- Patterns MUST conform to CEED Path Pattern Profile 1.0, identified by ceed-path-profile.json in the schema manifest for Public Review Draft 2. Version 1.0 does not depend on Git wildmatch or another library-specific wildcard implementation.
- Version 1.0 permits exactly two forms: an exact repository-relative path, such as src/services/ticket_service.py, and a directory-subtree pattern ending in /**, such as src/services/**. A directory-subtree pattern <base>/** matches paths strictly beneath <base>; it does not match <base> itself.
- No other wildcard syntax, character class, negation, leading slash, trailing slash, backslash, repeated separator, . or .. segment, or Unicode control character in the ranges U+0000 to U+001F, U+007F, or U+0080 to U+009F is valid. Pattern strings and changed paths MUST be Unicode NFC. Invalid patterns block issuance; invalid changed paths fail closed as invalid-path.

6.2 Classification

Apply one precedence rule to every changed path so the resulting approval has a repeatable meaning.

Every valid changed path, except the candidate's own verified Delivery Contract path under Section 6.3, MUST receive one final scope classification: authorized, protected-scope, out-of-allowed-scope, or policy-protection. A path may match both allowed_scope and protected_scope, but its final classification is exclusive. One merge evaluation may still record several failed rules when different paths fail differently or evidence is missing.

1. Apply repository-policy protections first. A matching repository-policy protection produces policy-protection, always blocks, and cannot be overridden by a Delivery Contract. A policy pattern that does not conform to CEED Path Pattern Profile 1.0 is an evaluation-level blocking error and MUST fail closed as policy-protection.
2. Collect every allowed_scope and protected_scope pattern matching the path.
3. If allowed and protected patterns both match, compare specificity. Specificity is twice the length of the literal path represented by the pattern, plus one for an exact path and zero for a directory-subtree pattern. The greater score determines the final classification.
4. A Delivery Contract MUST NOT contain the same pattern string in both allowed_scope and protected_scope. That duplicate is an equal-specificity contradiction and MUST be rejected at issuance with specificity-tie. If an invalid Delivery Contract nevertheless reaches evaluation, the defensive rule is that an equal-specificity allowed/protected match resolves to protected-scope and the candidate blocks.
5. If only protected patterns match, the final classification is protected-scope and the candidate blocks.
6. If only allowed patterns match, the final classification is authorized. If neither set matches, the final classification is out-of-allowed-scope and the candidate blocks.

6.3 Change types and repository edges

Check repository edges explicitly so renames, links, generated files, and submodules retain their stated boundaries.

- Every path in the Git diff MUST be checked regardless of file content or change type. The sole exception to ordinary scope classification is the candidate's own canonical Delivery Contract path after it passes the issued-version verification in Section 4.3. That path is never granted through allowed_scope.
- A rename contributes both the old path and the new path. A delete contributes the deleted path. Each path retains its own final classification. No single rename-level classification or severity roll-up is normative.
- Generated, binary, vendored, lock, migration, and build files are checked when Git reports them in the candidate diff.
- A symbolic link is checked by its repository path. The target is not followed or treated as another repository path. Authorization of a link path therefore does not authorize effects produced by following its target.
- A submodule entry is checked as one path in the parent repository. Changes inside the submodule require a separate adoption in that repository.
- Sparse checkout, local workspace state, and ignored uncommitted files do not change the candidate diff that the Router evaluates.

7 Candidate identity and merge enforcement

Merge integrations tie approval to the exact compared revisions. A changed base or head then receives a fresh decision.

7.1 Exact candidate

A merge-check decision is valid only for the exact candidate named in the entry. A rebase, force-push, base change, squash, or new queue construction creates a different candidate and requires a new merge-check.

7.2 Merge-check algorithm

The merge check follows these steps:

1. Validate the Delivery Contract against ceed-delivery-contract.schema.json, identified by the schema manifest for Public Review Draft 2, and verify that the referenced Delivery Contract version was issued under the applicable policy.
2. Read the changed-file list from Git for the exact base_sha and head_sha. The Router does not infer the list from a prompt or narrative description.
3. Apply the path authorization algorithm to every changed path.
4. Read the named evidence results from the configured CI providers through authenticated acquisition. The integration MUST establish the source principal and retain provenance sufficient to support that identity; a name, digest, or signed wrapper alone is insufficient. Evidence MUST NOT be represented as independently produced when the known control relationships contradict that claim. Every merge-evidence item MUST carry candidate_ref containing both base_sha and head_sha, and both values MUST equal the candidate under evaluation. Head-only evidence binding is not conformant.
5. Record all failed rules. One evaluation may fail more than one rule.
6. Write one merge-check entry for the evaluation, using Section 11's decision identity and retry rules, before returning approved, review-required, or blocked. Section 19 identifies the open conditions for selecting review-required.
7. The Git platform MUST treat the Router result as a required status check before merge where the commit exists before landing.

Implementation note (non-normative): enumerate the complete changed-file set from the exact Git objects. Representational State Transfer (REST) API file listings are convenience metadata. Include both rename paths. If completeness or exact batch attribution cannot be established, fail closed to single-change merges (Section 8.3).

7.3 Final-SHA rule

The source revision used to build a deployment artifact MUST itself have an approved merge-check. A merge queue normally creates and evaluates the prospective final commit before the default branch moves. A squash or rebase merge may create the final commit only as it lands; in that case the commit is evaluated after landing and the deployment gate holds the artifact until the final SHA has its own approved entry. For a post-land check, base_sha MUST be the landed commit's first parent.

A failed post-land final-SHA check prevents deployment. The organization MAY quarantine or revert the landed commit through its existing Git process. That remediation mechanism is outside CEED, but the blocking decision remains in the Change Passport.

8 Batch admission and combined commits

Queue integrations attribute each changed path to one Contract so combining work preserves each change's authorization.

8.1 Admission before construction

Where a queue exposes pending candidates before group construction, the Router SHOULD run the admission comparison at that point so incompatible changes never enter one group. Where no pre-construction hook exists, the Router MUST run the same check against the constructed prospective group before it can merge. A constructed group may therefore be discarded by the queue; the CEED Framework does not claim that every platform can avoid that cost.

8.2 Defensive merge-gate floor

At the merge check for a constructed batch, the Router MUST evaluate the constituents as they actually exist. The constituent file_subset values MUST be pairwise disjoint, their union MUST equal the complete changed-file set of the combined commit, and every changed path MUST be attributed to exactly one constituent. A path attributed to more than one constituent is an overlap and records file-overlap during batch-admission-check or late-overlap at the constructed merge-gate floor. A path present in the combined commit but attributed to no constituent, or duplicated within one constituent file_subset, records unattributed-path. Where overlap and unattributed-path conditions coexist in one group, the overlap reason takes precedence. Only constituents whose subsets participate in an overlap receive the overlap violation; the group remains held until the conflict is removed.

8.3 Integration contract

A batch integration supplies both required capabilities:

- The Git platform integration MUST expose the prospective group commit, immutable constituent identities, and exact changed-file subsets sufficient to prove disjoint and complete attribution.
- The integration MUST provide a path from a blocked group to the same changes merging separately: automatic requeue, author requeue, or batch size one.
- If either required capability is missing, configuration MUST fail closed to single-change merges.

8.4 Combined commit and deployment

Combining commits does not combine their Delivery Contracts into one authorization. Every constituent remains governed by its own Delivery Contract and Change Passport. The combined commit is decomposed into per-change candidates using the exact file_subset attributed to each change. The deployment gate enumerates constituents from the commit, confirms complete and disjoint attribution, then confirms one approved merge-check for every constituent in its Change Passport.

Combined constituents are release-compatible only when their approved strategy identifiers and ordered checkpoint-name sequences are identical. Required release evidence is the union calculated separately at each matching checkpoint. A mismatch SHOULD be caught during admission and MUST block at the constructed-group or deployment check if it remains.

9 Deployment enforcement

Release teams connect the artifact, deployment event, strategy, and checkpoint evidence so every approval concerns the rollout actually advancing.

9.1 Deployment identity and artifact binding

The Router assigns deployment_ref when a deployment event is created for an artifact built from an approved candidate. Where the configured provider exposes a stable authenticated identifier for the deployment event, the Router MUST either derive deployment_ref from it or record an auditable binding to it. Every deployment-check and production entry for that event carries the same deployment_ref. The artifact object MUST carry its digest and built_from_sha.

9.2 Ordered checkpoints

The release rule is an ordered checkpoint list, and checkpoint names within one resolved release rule MUST be unique. Checkpoint approval requires that every earlier checkpoint in the Delivery Contract holds an approved deployment-check entry for the same deployment_ref, Delivery Contract version, and artifact digest. Applied to the final checkpoint, this rule establishes sequence completion without a separate terminal state.

9.3 Strategy attestation

Each resolved release rule MUST mark exactly one checkpoint with strategy_verification_required: true. That is the first checkpoint at which the approved strategy governs production exposure. At that checkpoint the Router MUST receive an authenticated assertion of the strategy value, verify its association with the deployment event, and compare the value as text with the Delivery Contract value after any versioned policy mapping. Missing or mismatched assertion blocks. The deployment-check entry's release_strategy field records the authorized value from release_rule.strategy. The field name strategy_attestation does not imply cryptographic signing unless the implementation's security profile requires it.

The authenticated strategy assertion MUST identify, or be demonstrably bound to, the same authenticated deployment event represented by deployment_ref. strategy_attestation.reference identifies the provider record or authenticated assertion. strategy_attestation.evidence is a REQUIRED evidenceReference resolving to typed deploymentEvidence from which a third party can re-establish the same-event association. The evidence MUST bind the provider namespace, account or cluster, and event identity to deployment_ref, preserving authenticated source records and the exact field relations used to join them. Repeating deployment_ref without a provider binding is insufficient. If the association cannot be established, the checkpoint MUST block and MUST NOT be recorded as approved.

Evidence MAY arrive through more than one authenticated call to the same provider if every response is bound to the same deployment event. deploymentEvidence records the assertion_reference, authenticated subject and authentication, consumed_at, strategy_field, provider_native_value, association, basis_evidence, mapping, payloads, and acquisition. The primary assertion is the authenticated assertion that carries the strategy value. Each consumed assertion or response has its own payload_hash; the primary assertion's digest MUST match strategy_attestation.payload_hash. Retained payloads use encoding: base64 and bytes; digests cover decoded exact bytes, not the JSON wrapper. Every payload needed for a claimed reproducible observation MUST remain retrievable with its matching digest. A later provider response need not reproduce those bytes.

acquisition.kind is direct or adapter. When it is adapter, acquisition MUST include adapter_id and an immutable adapter_version identifying the deployed build by SHA-256. Direct acquisition carries no adapter identity or version. Adapter provenance records which component acquired and bound the evidence; it does not confer authority or promote a workload assertion to provider-record. mapping records either identity

mapping or a protected, versioned policy mapping with exact source and target values. Platform-native strategy mapping MUST NOT be hardcoded in Router logic, and a strategy MUST NOT be inferred from numeric traffic weights.

strategy_attestation MUST include subject, payload_hash, basis, and evidence in addition to attested_value, provider, reference, and matches_contract. The Router supplies subject as the principal authenticated at retrieval or signing; it is not necessarily the deployment triggerer or executing controller. The Router computes payload_hash as a SHA-256 digest of the exact bytes consumed. matches_contract alone is not evidence of equality; the Router compares the attested and authorized strategy values directly after permitted mapping.

basis has two permitted meanings. provider-record means the deployment platform supplied the strategy value through its own authenticated interface and bound it to that specific deployment event durably enough to answer for that event after current configuration changes. Merely returning the current strategy through an API is insufficient. self-asserted means the identified, authenticated deploying workload supplied a statement about its own deployment. These are established provenance facts, not claims that execution itself was independently proven.

The Router MUST block if the mapped strategy mismatches, no basis can be established, the issuance-bound risk tier and policy identified under Section 4.3 lack a valid non-empty release_baseline.acceptable_attestation_basis list, or the obtained basis is not in that list. Missing, null, or unknown basis values are rejected. There is no silent default or implied ranking between basis values. Recording self-asserted evidence as provider-record is nonconformant. Legacy strategy_attestation.hash MUST be rejected whether present alone or alongside payload_hash; it MUST NOT be migrated or reinterpreted as payload_hash.

The payload_hash rule also applies to outcome_source in Section 10.2: it records the exact consumed bytes and does not promise later provider reproducibility. outcome_source.evidence uses the same evidenceReference shape and is required when reproducible audit is claimed. Missing retained bytes make that audit claim unavailable; they do not by themselves create a universal deployment block. Same-event association remains mandatory at the checkpoint. No payload normalization method is defined; digests identify the exact consumed bytes.

9.4 Release evidence

Every release-evidence item MUST name the deployment_ref and checkpoint for which it was produced. It MUST declare one binding class: artifact or deployment-checkpoint. Artifact-bound evidence may travel with the same artifact. Deployment-checkpoint evidence MUST NOT be reused for another deployment attempt. Deployment-checkpoint evidence MUST NOT be reused at another checkpoint. Artifact-bound reuse MUST preserve the identical artifact digest. The integration MUST authenticate the configured evidence source and retain its provenance. Source identity, subject binding, and claims that the evidence was produced independently of the code-change producer MUST remain distinct; intermediary authentication alone establishes none of the upstream production facts. Section 19 identifies the open rule for retaining original context during artifact-bound reuse.

9.5 Required integration

The deployment system MUST wait for each required decision before continuing. A workflow that does not call the Router, or calls it without waiting, is not conformant. The Router does not execute the rollout; the deployment platform does.

10 Production outcomes and evolving knowledge

Operations teams append outcomes and later findings to the same deployment history so incident corrections preserve the original decisions.

10.1 Fresh production observation

After the observation window selected from the issuance-bound policy and risk tier under Section 4.3, the Change Passport writer appends one fresh production entry for the deployment. The status is healthy, incident, or inconclusive. The entry also records rolled_back, checkpoint completeness as complete/incomplete/unknown, and outcome_source.

An initial incident MUST carry incident_ref and incident_opened_at. A fresh production entry carries no reason code. More than one fresh entry for one deployment_ref is invalid_source_data unless the duplicate is byte-identical and is removed by idempotent retry handling.

10.2 Outcome provenance

outcome_source MUST record mode, provider, authenticated subject, retrievable reference, and payload_hash. Automated and manual outcomes remain distinguishable. The subject identifies the principal authenticated at retrieval or signing, not necessarily the person who triggered the event or the controller that performed it. The Router computes payload_hash as an algorithm-qualified SHA-256 digest of the exact bytes it consumed. The digest identifies those observed bytes; it does not preserve them or promise that later retrieval from a mutable provider will reproduce them. A reproducible-audit claim MUST be supported by outcome_source.evidence resolving to typed deploymentEvidence and by retrievable retained payloads for every response needed to reproduce the observation. No payload normalization method is defined; unversioned normalization is not permitted. The Router trusts authenticated metadata from the configured provider; it does not prove the internal integrity of that provider or runner.

10.3 Late incident association

A later incident is appended as a production entry with reason late-incident. It carries the earlier deployment_ref, a required incident_ref, and incident_opened_at copied from the organization’s incident record. The same real incident may be associated with several Change Passports. No Change Passport entry is declared the primary cause.

10.4 Retraction

A later analysis may clear an association. A compensating production entry with reason incident-association-retracted carries the same deployment_ref and incident_ref and a retrievable evidence reference. The original association remains. Effective association state is the latest appended entry for the pair, using append order rather than wall-clock time. Retraction requires the same authority as association.

A new association after a retraction uses reason late-incident and a new decision_id under Section 11. A retry of an earlier association or retraction does not change append order or effective association state.

10.5 Corrective cycles and new changes

A corrective cycle remains in the original Change Passport only when its sole purpose is to restore the intended outcome of that original change. It receives a new Delivery Contract version and passes through merge, deployment, and production again. A fix spanning several original changes, introducing independent intent, or governed through a separate work item MUST receive its own Delivery Contract and Change Passport. Relationships among such changes remain in the organization’s work and incident systems.

10.6 No terminal state

The Change Passport has no closure event and no terminal status. Each entry records what was known and decided at its timestamp. A later entry may change the effective classification without changing any earlier entry. Reports are therefore as-of views and may legitimately differ when regenerated later.

11 Change Passport store and integrity boundary

Store implementers protect authoritative writes and retain readable history so retries, later checks, and reports use consistent records.

Authoritative Change Passport entries MUST be appended through an authenticated, explicitly authorized protected writer identity. Producer execution credentials MUST NOT grant append authority, access to writer credentials, or control of the writer or store access policy. Authorized read access, including access to rejection explanations, remains permitted. The writer MUST enforce append authorization before accepting a write. Protection has the meaning in Section 16.2; the human-administrator trust limit in Section 16.1 remains applicable. The store follows these recording and retention rules:

- The order of objects in the entries array is the authoritative append order. The value of the entry field MUST equal the object's one-based array position, beginning at 1 and increasing by exactly 1 without duplicates or gaps. A Change Passport that violates this invariant is invalid_source_data.
- The store MUST support consistent reads for checkpoint sequencing and effective-state calculation.
- Retries for the same logical decision MUST be idempotent. Logical identity is the tuple defined below. A retry with the same tuple and byte-identical decision payload returns the existing entry. The decision payload contains every entry field except the store-assigned entry number, including the original timestamp and evidence references; file_subset is normalized as specified below before recording. Returning a stored entry acknowledges the earlier write; permission for a later governed action still requires the applicable checks. The same tuple with different decision content is invalid_source_data and MUST NOT append a second conflicting decision. For a conflicting contract-review-rejected retry, the Router MUST preserve the first record, append nothing, and leave the rejection count unchanged; the attempt is blocked.
- Logical decision tuples: contract-review-rejected = Change Passport UUID + entry_kind + review_id; contract-issued = Change Passport UUID + entry_kind + contract_version; batch-admission-check, merge-check, and deployment-check = Change Passport UUID + entry_kind + decision_id; fresh production = Change Passport UUID + entry_kind + the literal fresh + deployment_ref; incident association or retraction = Change Passport UUID + entry_kind + decision_id. Candidate, Contract, batch, deployment, checkpoint, artifact, incident, and reason bindings remain part of the decision payload; changing them under the same tuple is a conflict. Where file_subset is present in a decision payload, implementations MUST compare it as a set and MUST normalize it to a sorted sequence before recording and comparison.
- Each batch-admission-check, merge-check, deployment-check, and reason-coded production entry MUST carry decision_id, a non-empty opaque string identifying one evaluation or incident event within its entry kind and Change Passport. The Router or authorized incident integration MUST assign and retain that identifier before recording the decision and preserve the identifier and completed decision payload across retries and restarts. A stable source evaluation or event identifier may be reused if it satisfies that uniqueness scope. A new evaluation, including one after new evidence, completed review, or a changed candidate, receives a new decision_id and appends a new entry even when the evaluated subject is unchanged. A retry reuses the original identifier and saved decision payload; it does not re-evaluate the subject. Identity lookup, payload comparison, and append MUST be atomic for one logical decision. A new identifier grants no authorization and does not replace any required check.
- The store MUST support a consistent snapshot across every Change Passport selected for one report. The report records snapshot_ref. A deployment_ref shared by several Change Passports MUST be evaluated from the same snapshot.

- The CEED write interface MUST NOT provide update or delete operations for existing entries.
- Every Delivery Contract version referenced by a retained Change Passport MUST remain retrievable for as long as the Change Passport is retained. Ordinary Git reachability alone does not establish durability after branch deletion, history rewrite, or force-push. The implementation MUST substantiate retrieval over its retention period; no particular protected ref, archive, or store is mandated here. Supporting authorization evidence referenced under Section 4.3 MUST remain retrievable for that period. Where reproducible audit is claimed, the exact consumed payload evidence MUST also be retained.
- Retention duration is an organizational policy and is not fixed by CEED Version 1.0.

Append-only describes the record and write interface. Optional signatures, immutable storage, and independently anchored integrity records strengthen retained history without requiring one store or cryptographic design (Section 16.4).

An implementation MAY encode selected Delivery Contract or Change Passport statements as in-toto supply-chain attestation predicates, use Dead Simple Signing Envelope (DSSE) to carry signed payloads with or without in-toto, transport lifecycle notifications through CDEvents, a standard format for software-delivery events, or evaluate CEED rules through Open Policy Agent (OPA) or another policy engine. Such bindings are optional. They MUST preserve CEED identifiers, append order, decision meaning, and reporting semantics; the external substrate does not replace the normative CEED record.

12 Change Passport-native local reporting

Reporting teams calculate local measures from retained Passports so the organization can inspect repeated authorization and delivery patterns.

12.1 Operation

A Change Passport-native reporter provides a local reporting command. It reads a consistent snapshot of one or more local Change Passports, calculates the Version 1.0 measures across the selected cohort, writes an inspectable report file and digest, and produces a local view or export. Report generation MUST perform no network access and MUST remain outside the merge and deployment critical path.

12.2 Scope boundary

Local reports measure recorded authorization, progression, batches, and production history. They cannot find actions with no relevant Passport entry. Organization-wide denominators need later read-only repository, release, and incident inventories.

Public exports of reports MUST follow the declaration-exclusion rule in Section 18, “Schema package and test vectors.” Internal coverage inspection does not extend the Passport-native reporter beyond its defined inputs.

12.3 Determinism and as-of semantics

The following rules define reproducible report inputs and results:

- The report MUST carry report_schema_version, metric_definition_version, a Coordinated Universal Time (UTC) period with start_inclusive and end_exclusive, generated_at, source, and metrics.
- The source object MUST state type ceed-change-passports, network_access false, and the Change Passport-store snapshot_ref used for the calculation.
- Every rate MUST publish numerator and denominator. A value MAY be included as a convenience but MUST equal numerator divided by denominator.

- Every absence MUST use the fixed missing-status vocabulary in Appendix B rather than an unlabeled null.
- A report is a snapshot as of generated_at. Regeneration may change an earlier period when later Change Passport entries add knowledge.
- The reporter MUST validate input schema versions and mark unsupported records rather than guessing. Records with invalid append order, duplicate or conflicting logical decisions, duplicate or conflicting fresh outcomes, missing contract-issued entries required to govern subsequent lifecycle entries, or unsupported schema versions are excluded from metric cohorts and counted in data_quality using invalid_source_data or unsupported_schema_version. A structurally and semantically valid rejection-only Change Passport is valid source data even without a contract-issued entry. It MUST NOT be counted as invalid_source_data and contributes to no issuance or merge cohort. Rejection entries do not count as issued Delivery Contract versions or merge evaluations; Appendix A defines the cohorts.

12.4 Initial metric families

Family	Version 1.0 measures	Use
Authorization	changes evaluated; merge evaluations; authorization block rates; allowed and protected-scope violations; missing evidence; review required	Local report and initial shared comparison
Change progression	30-day protected touches; 30-day retry density; Delivery Contract versions; risk-tier distribution	Local report and selected context
Batch integration	batch admissions; admission conflicts; late-overlap blocks	Local diagnostics; late-overlap may be compared
Recorded production history	recorded deployments; initial outcomes; rollback flags; late incident associations; retractions	Local only until independent denominators exist

Table 5. Local reporting families. Passport records support authorization, progression, and batch measures; production history stays local until independent denominators exist. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

The metric definitions for the Version 1.0 Change Passport-native profile are in Appendix A.

13 AI Delivery Practice Profile

Organizations preparing optional aggregate contributions (Section 15) describe how people and AI participate at each delivery stage so comparisons use declared operating context.

The AI Delivery Practice Profile records contributor-declared context at each stage. The reporter derives the practice band (the operating-pattern category defined in Section 13.1) from those declarations, rather than independently verifying them. The band describes an operating pattern, not maturity, quality, safety, or autonomy.

Field	Allowed values
requirement_formation	human-led; ai-assisted; agent-drafted-human-approved
contract_drafting	human-authored; ai-assisted; agent-drafted-human-approved; policy-derived
contract_authorization	human-approved; human-approved-with-ai-recommendation; delegated-automated
code_generation	human-written; ai-assisted; agent-executed
test_generation	human-written; ai-assisted; agent-executed
evidence_collection	manual; automated
merge_authorization	human-decided; machine-recommended; machine-enforced
deployment_authorization	human-decided; machine-recommended; machine-enforced
production_outcome_recording	manual; automated
exception_and_scope_decisions	human; human-with-ai-recommendation

Table 6. Declared delivery practice by stage. Stage values describe where people and AI participate; the reporter derives the practice band defined in Section 13.1 from those declarations. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

policy-derived records deterministic generation from authoritative requirements and protected policy. delegated-automated records route B issuance: the contract-issued entry requires decision_method with ai-review or mechanical-only and carries authorization_evidence. Separate AI review is the normal path; protected policy and Router classification determine the narrow mechanical-only exception regardless of producer. Route A uses human-approved with approver and authorization_evidence and forbids decision_method. Human authorization values do not permit producer self-approval. The context profile and the inline profile in the contribution schema use contract_authorization. The policy field remains contract_approval.

The optional ai_use object contains only contract_authorization, merge_authorization, and deployment_authorization, each with present, none, or unknown. contract_authorization applies only when that stage is delegated-automated; merge_authorization and deployment_authorization apply only when the corresponding stage is machine-recommended. An absent applicable declaration means unknown, never none. Inapplicable keys and contradictory declarations are invalid. For a stage that mixes AI and non-AI use, the contributor declares present. Known AI review cannot be declared none. These declarations for the delivery unit (whose boundary remains open in Section 19) MUST NOT be inferred automatically from one Change Passport.

13.1 Derived band

Derive the band from the declared stage values so two reporters classify identical context alike.

Band	Derivation
human-led	No AI is used in the declared profile. Deterministic automation is allowed, including policy-derived drafting, mechanical-only delegated issuance, and non-AI machine recommendations. Applicable ai_use declarations are none, and no higher-band condition applies.
ai-assisted	An explicit AI stage value or an applicable ai_use value of present establishes AI assistance, and neither ai-native nor agent-led applies. AI markers include ai-assisted, agent-drafted-human-approved, human-approved-with-ai-recommendation, and human-with-ai-recommendation on exception and scope decisions. machine-recommended and delegated-automated alone do not establish AI use.
agent-led	Code generation or test generation is agent-executed, but the full ai-native rule is not met. This band does not imply human merge gating.
ai-native	code_generation is agent-executed; evidence_collection automated; merge_authorization and deployment_authorization machine-enforced; production_outcome_recording automated; exception_and_scope_decisions remains human or human-with-ai-recommendation. Humans may still write requirements, Delivery Contracts, and tests; those activities do not disqualify this band.
unclassified	The profile has an unsupported version, a missing or invalid required stage, or a malformed or contradictory declaration; or an applicable ai_use declaration is missing or unknown and no established higher band applies. Deriving unclassified does not make malformed source data valid.

Table 7. Deriving the declared practice band. The fixed predicates assign the same band to identical stage declarations without implying quality or safety. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

The Change Passport-native reporter MUST compute derived_band in this order: (1) return unclassified for an unsupported version, missing required stage, invalid stage, or malformed or contradictory declaration, while still reporting malformed source data; (2) apply the full ai-native predicate; (3) otherwise return agent-led if code or test generation is agent-executed; (4) otherwise return ai-assisted for any explicit AI stage value or applicable ai_use value of present; (5) otherwise return unclassified if an applicable ai_use declaration is missing or unknown; (6) otherwise return human-led. Unknown AI use at a lower-priority stage does not erase an already established higher band. If a supplied derived_band differs from the calculation, the profile is invalid_source_data. Local and shared views MUST label the result as declared context.

14 Toolchain Context Profile

Record each tool’s delivery role so a comparison describes what the tool did in that workflow.

The Toolchain Context Profile records each tool’s lifecycle function and role using the named ceed-tool-catalog.json version. A local interface may present those choices as checkboxes.

Lifecycle function examples	Role values
requirement-management; contract-authoring; code-generation; source-control; continuous-integration; static-analysis; security-analysis; test-automation	system-of-record; draft; execute; evidence-provider; enforce
artifact-provenance; policy-evaluation; merge-control; deployment; traffic-management; observability; incident-management; reporting	system-of-record; evidence-provider; enforce; deploy; observe; report

Table 8. Toolchain roles. Lifecycle functions and roles describe each tool’s contribution to delivery. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

Toolchain entries follow these rules:

- Each tool entry MUST carry lifecycle_function, product_id, and at least one role. The reporter MUST validate product_id and lifecycle_function against the named catalog version.
- A product not present in the catalog uses other:<local-id>. The raw local identifier is not published automatically.
- Product versions MAY be retained locally. Version-level public cohorts are not part of Version 1.0.
- Capability-level comparison is the default public use. Product-level comparison is permitted only when enough independent organizations satisfy the privacy rules.
- Exact-stack cohorts MUST NOT be published because sparse combinations can identify contributors and fragment the population.
- Toolchain relationships are observational. A difference between cohorts does not establish that a tool caused the difference.

15 Optional aggregate contribution

Organizations can prepare and inspect an aggregate contribution before a separate submission; this preserves local control over what leaves the organization.

15.1 Preparation and submission

Preparation and submission follow these steps:

1. prepare reads the local report and optional context, creates an inspectable contribution file and digest, and performs no network access.
2. submit is a separate explicit action. Before sending, it displays the destination, exact file, digest, and fields being sent.
3. Scheduled submission MAY be configured separately but MUST NOT be the default.

15.2 Data boundary

A Version 1.0 contribution MUST validate against ceed-passport-native-contribution.schema.json. It contains Change Passport-native delivery-unit aggregates, persistent private pseudonyms for intake deduplication, declared context, and profile versions. It MUST NOT contain Delivery Contracts, Change Passport UUIDs, paths, SHAs, batch references, deployment references, incident references, source code, requirements, or individual entries. Published files contain cohort aggregates only and exclude pseudonyms and delivery-unit rows.

An actual Enforcement Coverage Declaration, its validation view, and its history MUST NOT be included in a contribution or public export (Sections 16.2 and 18).

The derived band remains the headline grouping for AI delivery practice. Published comparisons MAY use one profile field at a time under the same Section 15.3 sample-size, concentration, differencing, and suppression thresholds. Full combinations of the ten stage fields MUST NOT be published. Neither a band nor a one-field comparison is an autonomy or maturity score.

15.3 Publication thresholds

The following thresholds govern public cohort cells:

- A cohort cell requires at least 10 delivery units from at least 5 independent organizations. The delivery-unit definition remains open (Section 19).
- Each organization’s share is at most 25 percent of the units and at most 25 percent of the relevant event denominator.
- Parent and child cells MUST be checked for differencing risk and complementary suppression applied where required.
- A suppressed rate, numerator, and denominator are all replaced by insufficient_sample. The failing threshold is not disclosed.
- At launch, broad cohorts are preferred over weakened privacy thresholds.

Optional participation produces observational comparisons. Self-selection and uncontrolled differences limit causal conclusions; a cohort difference identifies a question to investigate.

16 Security and operational considerations

Security and platform teams use these boundaries to protect delivery authority and evidence from the AI agent making the code change.

16.1 Threat model and trust boundaries

The Delivery Contract bounds what a code-producing AI agent can get approved at connected gates. Protected policy, evidence acquisition, and Passport writing keep authority and records outside its execution permissions. An AI producer acting against its instructions may submit misleading proposals, edit its files, coordinate with other agents, or seek another route. Privileged provider, runner, administrator, or credential compromise is a separate threat; shared control can defeat apparent service separation. The design relies on the following trust boundaries:

- Provider trust: the Router consumes authenticated metadata from Git, CI, deployment, observability, and incident providers. It does not independently prove provider or runner integrity.
- Human-administrator trust limit: a human who controls policy, issuance credentials, or the protected review waiting limit is a trusted governance actor. The Change Passport shows that the change followed the configured process; it does not establish independence from that administrator. Neither a bot account nor AI review supplies that independence. The human route B allowance does not authorize bypass, self-approval, or omission of the required checks and evidence.

16.2 Enforcement Coverage Declaration

Before relying on CEED, the organization records where its delivery actually waits for CEED decisions, which bypass routes exist, and which properties were verified. The internal declaration gives owners a map of gaps to close; keeping it private protects that map.

A deployed arrangement MUST maintain an Enforcement Coverage Declaration before first relying on the CEED Framework for a governed action or making an operational coverage claim, whichever occurs first. The requirement applies to the adopted delivery scope, including staging or test environments used operationally. Isolated components and wholly synthetic fixtures do not require an Enforcement Coverage Declaration. One bounded declaration may cover several repositories.

The declaration MUST identify its format revision, the arrangement through a stable identifier, adopted component or composite scope, repositories and refs, governed actions, target environments, and enforcement points. For each claimed connection it MUST distinguish mandatory enforcement, advisory use, absence, and unknown coverage. It MUST state the inventory boundary, exclusions, unknowns, known bypass and emergency routes, and relevant unassessed dependencies. Connection alone is not enforcement; an empty exclusions list does not prove completeness.

Each coverage assertion MUST state whether it is declared only, verified for a stated test scope, failed, or unknown. Verification claims MUST identify the property checked, assessment date, method, assessor and control relationship, exact configuration, and protected supporting evidence. The declaration MUST identify an accountable human or organizational owner through a resolvable identity. It MUST record its assessment date, revision, predecessor revision or explicit initial revision, and the period to which that revision applies. An owner is not automatically an independent verifier.

The organization MUST reassess the declaration after relevant changes to scope, gates, policy, enrollment, credentials, writer or store protection, evidence providers, integrations, bypass or recovery routes, or facts underlying verification. Reassessment follows an organization-defined procedure evidenced by inspection. Prior revisions MUST be retained with their applicability periods. A stale verification MUST NOT support a current claim. Declaration reassessment requires no per-action lookup or automated transition check and produces no authorization result.

Protected means access is authenticated and explicitly authorized, authoritative changes are attributable, and retained records and their access policy are outside producer execution control. Producer execution credentials MUST NOT create, approve, replace, or delete authoritative declaration revisions or relax their protection. Authorized internal reading remains permitted. A trusted human governance administrator may own the declaration; combined governance and producer roles MUST be disclosed truthfully under the human-administrator trust limit in Section 16.1. No storage product or universal second-person approval is prescribed.

The declaration MAY use any internal format, but MUST provide a small protected JSON validation view of all required content. A generated view MUST identify the exact source revision by protected reference and byte digest, with correspondence established. The declaration, view, and history MUST remain internal and MUST NOT be published. The separate validation schema and wholly synthetic examples may be published under Section 18.

Declared scope MUST NOT waive a required gate within the adopted scope. An absent, advisory, excluded, or unknown gate cannot make a governed action conformant or CEED-approved (Sections 7.2 and 9.5). Missing or stale declaration evidence prevents a conformant current deployment-coverage claim; it does not replace action authorization.

16.3 Authorization and evidence boundaries

Keep the source of permission separate from the source of a proposed Contract so a trusted intermediary cannot turn a draft into authority.

A Contract proposal gains authority through Section 4's authorization process, with authenticated input sources, protected context selection, exact commit references, and retained evidence. Repeating a proposal through a trusted workflow grants no authority. On route B, a human producer's control of policy or issuance credentials is recorded as an organizational declaration, without a claim of independence.

Evidence acquisition establishes who supplied a record; subject binding establishes what it concerns; its basis states what that source can support. Sections 4.3, 7.2, 9.3–9.4, and 10.2 govern those separate obligations. A digest establishes byte identity, and a signature may authenticate a signer; neither proves truthful execution or independent custody. Truthful self-asserted strategy evidence remains permitted where protected policy accepts it. Adapters and security findings receive no authority beyond the applicable authorization process.

Path authorization makes the approved boundary enforceable; software security determines whether the implementation inside it is acceptable.

16.4 Passport integrity and optional protections

Protect Passport writes so investigators can follow retained decisions without granting the coding agent control of its history.

Section 11 protects writer access, append order, consistent reads, safe retries, and retention. An authorized writer may be any conformant implementation. A privileged storage administrator remains inside the trust boundary.

Organizations may strengthen retained records with signatures, immutable storage, write-once retention, or integrity tools. A hash chain needs independent anchoring outside the writer's control to expose a rewritten history. These optional protections can reveal alteration; they establish neither event truth nor completeness and remain unverified for the CEED Project implementation.

16.5 Availability and bypass at connected gates

Require the merge and deployment systems to wait for gate decisions so the checks control actual progress. The following boundaries govern availability and bypass:

- Gate availability: a mandatory merge or deployment gate fails closed. A reporting failure never blocks delivery.
- Administrative bypass: a platform administrator may bypass native controls. A bypassed action MUST NOT be represented as CEED-approved, including retrospectively. The bypass MUST remain auditable in the authoritative platform record. If the Router was never invoked, it cannot record the bypass. Version 1.0 defines no Change Passport bypass event; observing and importing bypass information is possible future work.

Recommended recovery practice is for emergency and recovery owners to retain evidence and restore normal enforcement. Required gate decisions control progress only when delivery waits for them (Section 9.5). Compromised or bypassable dependencies can defeat enforcement; connected gates cover only the agent actions routed through them.

16.6 Integration and analytical uses

Security teams can connect lifecycle records to their investigation tools so signals lead to an accountable response.

Security platforms may consume authorization and lifecycle events, supply evidence or incident findings, or read the Delivery Contract for analysis. Reading supplies context; a mandatory gate makes delivery wait. Authorized processes evaluate incoming findings under Section 16.3; an integration cannot widen scope, suppress gates, or grant exceptions.

Independent production inventories can be compared with artifact and change records to find missing associations for investigation. Optional alerts and cross-change analysis can flag repeated blocks, retries, or correlated activity; authenticated outputs report signals, not hostile intent or permission. Section 4.3 governs the mandatory intervention alerts. These integrations await implementation verification.

Recommended operational practice is for organizations to restrict sensitive telemetry and name owners for investigation, containment, and restoration. Tests, builds, dependencies, and previews may execute code before merge; their execution environments and credentials need protection too.

16.7 Limits of enforcement and verification

Read verification results against the properties actually tested so a passing case supports a bounded conclusion. The path algorithm limits approval to the issued repository boundary. Security controls assess malicious behavior inside allowed files; the limits below identify what each verification result supports.

- Review judgment and authenticity: objection accounting connects review inputs to recorded responses; it does not establish their adequacy or the reviewer’s correctness. Removal of an exact path from the proposed authorization can be checked mechanically. The Conformance Pack does not establish whether a proposed correction is substantive or merely cosmetic. Authenticity depends on the reviewer integration, beyond byte integrity.
- Change lineage and late answers: finding the same proposal relabeled as a new change requires lineage evidence. Expired-review objections may be missed by a fresh review. Deadline rules establish neither predictable rejection nor selective suppression; exploitation depends on producer capabilities and reviewer behavior.
- Intervention availability: a hold waits for human resolution; expired answers cannot release it. Notification delivery affects whether handlers see an alert. Choosing "none" permits rejection cycles without numeric intervention; waiting limits, objection accounting, and all other authorization rules still apply.
- Executable coverage: the Conformance Pack exercises rejection records, objection accounting and counts, and represented transitions in synthetic sequences. Live exclusivity, deadlines and clocks,

restart, persistence, late-answer exclusion, intervention, notification failure, human release, resubmission, and atomic issuance still need integration verification. GitHub hold display, backup handler identity, release records, long holds, and producer access to explanations also need verification. These specified runtime requirements remain uncovered until the identified implementation's results enter the test coverage map.

- Evidence replay: deployment-checkpoint evidence is bound to one deployment_ref and checkpoint. Artifact-bound evidence reuse is limited to the identical artifact.
- Delivery Contract quality: a candidate staying within a Delivery Contract does not tell the Router whether the authorized scope is too broad. Risk-tier distribution and local review remain necessary.
- Requirement correspondence: the Router verifies that a candidate is governed by an issued Delivery Contract. It cannot verify that the change implements the requirement that the Delivery Contract names. Issuance review, code review, and testing remain responsible for that question. The issuance-context checks in Section 4.3 do not remove this semantic limit.
- Effect boundary: path authorization classifies repository paths in the Git candidate. It does not prove that code in an allowed path cannot create, modify, or influence a protected resource at build or runtime. Existing build isolation, sandboxing, testing, security analysis, deployment policy, and runtime controls remain responsible for those effects.
- Submodules: authorization of a parent-repository submodule pointer is not authorization of changes inside the referenced repository. That repository requires its own CEED adoption or equivalent controls.
- Sensitive data: Delivery Contract requirements, paths, identities, and Change Passport entries remain local unless the organization shares them directly. Declaration handling has the stricter internal-only rule in Section 16.2.

17 Conformance requirements

Reviewers use this matrix and the identified test snapshot to tie each claim to a component and its tested obligations.

Requirement	Delivery Contract producer	Change Passport writer and store	Gate implementation	Change Passport-native reporter
Valid schema package version	Required	Required	Required	Required
Exactly seven Delivery Contract fields	Required	Not applicable	Validated	Not applicable
Immutable Delivery Contract versions	Required	Recorded	Required	Observed
Exact candidate and final-SHA binding	Not applicable	Recorded	Required	Observed
Ordered deployment checkpoints	Produced	Recorded	Required	Observed
Append-only production knowledge	Not applicable	Required	Writes	Reads
Change Passport-native metrics	Not applicable	Supplies input	Supplies decisions	Required
No-network local generation	Not applicable	Not applicable	Not applicable	Required

Table 9. Selected conformance obligations by component. Contract creation, storage, gates, and reporting each contribute stated obligations to a full delivery claim. The full requirement set is defined throughout this Specification, including Appendix C. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

Non-normative legend. Required identifies a component obligation under its governing sections. Validated means the gate validates the Delivery Contract's seven fields. Recorded means the store retains the issued version, candidate identity, or checkpoint decision. Writes means the gate supplies production knowledge for the Change Passport writer to append; Reads means the reporter consumes that history. Supplies input means the writer and store provide the recorded data for reporting; Supplies decisions means the gate provides its recorded decisions as reporting inputs. Produced describes the ordered checkpoint plan supplied in the Delivery Contract's release_rule (Sections 2 and 5). Observed describes the reporter reading recorded Contract versions, candidate references, and checkpoint history to calculate its defined measures (Sections 11–12 and Appendix A). Not applicable means that this component has no separate duty for the named requirement. For the seven-field rule, the store records the issued version rather than validating the Contract's fields (Sections 4.3, 5, 7.2, and 11). Candidate and final-SHA checks belong to the gate, not Contract creation (Sections 2 and 7). These explanations apply only to the identified component and cells.

A conformance claim MUST identify the implementation build and builder, platform scope, Specification edition and revision, Conformance Pack edition and exact snapshot, component or composite scope, configuration, assessment date, results, and applicable requirements not tested. It MUST be tested with the identified pack's schema checks, semantic validation, and behavioral vectors. Conformance is to the Specification requirements, not merely to a passing runner. The coverage map MUST distinguish completed, uncovered, and provisional coverage (Section 18). Claims cannot imply that an uncovered requirement has been tested. A component claim MUST NOT be presented as a claim about a full delivery arrangement. A Conformance Pack for an earlier Specification edition does not validate Public Review Draft 2.

The intended coverage includes Delivery Contract structure; CEED Path Pattern Profile 1.0; normalization, change types, and path precedence; both authorization routes and the rules no route can skip; exact authorization and post-land binding; stable change identity, authoritative review and rejection records, objection accounting, rejection limits, intervention, and valid rejection-only reporting inputs; review lifecycle and atomic issuance; mandatory candidate ancestry and later versions; complete candidate and evidence binding; final-SHA first-parent identity; batch attribution and overlap; release compatibility and evidence union; strategy cardinality, same-event association, truthful basis, and exact-byte payload hashing; checkpoint binding and order; idempotency, new evaluations of unchanged subjects, re-issuance of identical Contract bytes, append order, fresh-outcome uniqueness, late incidents, retractions, and re-associations; issuance-bound attestation-basis and observation-window lookups; profile derivations, including applicable ai_use declarations; multi-Change Passport reporting; and unsupported schema versions. This list states required coverage areas, not a claim that they are all implemented in the current executable suite.

Existing security cases exercise supplied context and grant selection, declared control, exact candidate and event references, evidence basis, byte hashing, append order, safe retries, and represented review transitions. They establish fixture behavior, not live acquisition or custody.

Specified executable coverage includes declaration-view content and applicability, unsupported verification claims, no-waiver behavior with scope and policy held fixed, unauthorized declaration replacement, and exclusion from public exports. Writer-interface cases distinguish authorized appends from producer or unauthorized writes and preserve existing records after rejected writes, updates, or deletions. Issuance cases reject a trusted intermediary wrapping unauthorized producer input. Evidence-acquisition cases distinguish authenticated configured sources from wrong or unauthenticated principals, digest-only authenticity claims, and unsupported independence claims. Direct release-evidence cases reject a different deployment attempt or checkpoint and permit reuse only for an identical artifact where Section 9.4 allows it. These cases have uncovered status until registered assertions execute.

Organizations supply separate operational evidence for inventory completeness, credential custody, provider integrity, retention, and declaration reassessment. Reassessment, historical applicability, and current verification are inspection-only duties, uncovered by executable cases. A schema-valid view cannot establish them. Section 16.2 limits claims with missing or stale declaration evidence; component claims and Section 16.7 runtime gaps retain their own applicability.

18 Schema package and test vectors

Implementers pair schemas and tests with an exact Specification revision so structural checks and behavioral claims have an inspectable basis.

The manifests identify Public Review Draft 2 schemas, conformance artifacts, checks, and coverage. Exports use schemas/ and conformance/. Each document has its own publication date; the pack identifies its exact Specification revision and snapshot date. Machine schema versions and profile identifiers retain their separate identity. The package identifies these artifacts:

- ceed-delivery-contract.schema.json
- ceed-change-passport.schema.json
- ceed-policy.schema.json
- ceed-local-report.schema.json
- ceed-context-profiles.schema.json
- ceed-passport-native-contribution.schema.json
- ceed-tool-catalog.json
- ceed-tool-catalog.schema.json
- manifest.schema.json
- manifest.json
- ceed-path-profile.schema.json and ceed-path-profile.json
- conformance/conformance_manifest.json, coverage_map.json, vectors.json, run_conformance.py, schema_validation.py, named examples, report fixtures, and golden outputs

A separate Draft 2020-12 JSON Schema for the Enforcement Coverage Declaration validation view is specified. The schema manifest MUST register that schema; the conformance manifest and coverage map MUST register its wholly synthetic examples, applicable checks, and uncovered obligations. The validation-view schema and new checks are specified; implementation and execution remain to be evidenced in the coverage map. The validation schema and checks are separate from the Delivery Contract fields, Change Passport entry kinds, and conformance classes. The schema filename remains open (Section 19).

CEED Start Here, The CEED Framework Whitepaper, this Specification, CEED in Practice, CEED Conformance Pack Guide, and CEED Conformance Pack are identified by their Public Review Draft 2 editions. CEED Tool Verification Matrix is independently dated. Its Public Review Draft was published October 3, 2026; the tools were reviewed September 9, 2026. The Matrix’s correction notice is dated October 8, 2026. The documents, JSON schemas, path profile, tool catalog, and conformance materials are licensed under CC BY 4.0. The CEED Router implementation source is planned to be licensed under Apache License 2.0. Those licenses do not grant endorsement, official status, a right to use CEED trademarks outside applicable trademark rules, or a right to claim that the CEED Project has certified conformance.

Schemas check structure; semantic checks and behavioral vectors check invariants beyond JSON Schema: normalized paths, unique checkpoints, append order, immutable authorization, exact changes and evidence, complete batch attribution, rollout sequencing, one fresh outcome, and effective incident associations.

Required coverage for CEED Conformance Pack, Public Review Draft 2, includes the applicable authorization, evidence, and security rules in this Specification. Every nominally valid fixture, including reporting inputs, MUST be validated directly. Where executable coverage is incomplete, the pack MUST identify the specified rules and their schemas, executable tests for the completed subset, an explicit coverage map, and provisional schema parts labeled as such. Remaining behavior MUST be labeled “specified but not yet covered by the published executable suite.” The historical baseline may be preserved separately but MUST NOT be presented as validating this edition.

conformance/coverage_map.json MUST contain three lists: artifacts, checks, and requirements. artifacts identifies each packaged artifact or supported fragment, its schema target where applicable, intended validity, and checks. checks identifies executable checks, their input artifacts, and expected outcomes. requirements maps Specification requirements to their check identifiers and states completed, uncovered, or provisional coverage with any limitation. A list of check identifiers alone does not establish completed coverage;

completed coverage requires execution of the relevant assertions. Exported package references MUST resolve within the exported schemas/ and conformance/ layout. The coverage-map requirements inventory MUST include the requirements and verification limits in Sections 16.2 and 17. Inspection-only obligations MUST remain identified as uncovered by executable cases; a schema-valid example or a list of check identifiers is not operational evidence.

Public package and report exporters MUST exclude actual Enforcement Coverage Declarations, validation views, history, and their protected contents or references, including in attachments, nested data, diagnostics, and logs. They MUST accept only approved public sources; renaming a private record or labeling it synthetic is insufficient. Redacting an actual declaration does not make it wholly synthetic. Public declaration material is limited to the schema and separately authored wholly synthetic declaration examples. Authorized internal inspection remains permitted (Section 16.2).

18.1 Planned implementation scope

The CEED Project implementation targets Git and GitHub. Version 1.0 includes a GitHub deployment-identity adapter and no strategy adapter. Builders document their adapter contract and fixtures with the implementation. The Router authenticates and records adapter output as evidence, not authority. Universal platform support is not a publication requirement.

The Change Passport identifies the strategy evidence actually obtained. A provider-record claim needs measured event-specific evidence. Platform support claims identify repository visibility, owner type, plan, permissions, and configuration; an isolated platform test establishes only its tested behavior.

Teams may begin with issuance, exact-change and path checks, merge evidence, and Passport recording. The organization records that delivery scope in its internal declaration (Section 16.2). Version 1.0 also requires the CEED Project implementation to exercise deployment and local reporting. Conformance Pack fixture checks and implementation verification establish separate results.

18.2 GitHub implementation profile

GitHub integrators assess required checks, bypass, and human release against the configuration in use so platform behavior supports the claimed control. Section 4.3 defines this profile’s review-state selection, exact commit_id checks, App identity pinning, and CHANGES_REQUESTED rejection rules.

The GitHub implementation profile MUST prohibit the GitHub “Exempt” bypass mode for every identity able to change governed refs. The scoped assessment result is met, not-met, or unavailable. A proven grant of the “Exempt” bypass mode to an identity able to change an in-scope ref yields not-met even when other evidence is incomplete. Without a proven violation, unreadable or incomplete relevant configuration, an omitted bypass list, unresolved actors, or an incomplete inventory of identities able to change the refs yields unavailable. met requires complete readable evidence with no prohibited intersection, for the assessed scope and observation time. An omitted list MUST NOT be treated as empty. The GitHub “Always” bypass mode remains possible, visible, and logged; a bypassed action is never CEED-approved. The authoritative platform record MUST remain available as required by Section 16.5.

For the measured GitHub rule-suite surface, detecting bypass requires reading suite details, including suites marked pass: a suite result of bypass or any rule-level skipped result identifies a bypass. The record distinguishes a refused attempt from a completed bypass; neither a successful merge response nor a suite-level pass establishes approval. The GitHub App permission “Administration: read” was sufficient for the named public rule-suite list and detail calls tested on September 17, 2026. This finding does not resolve write-level bypass-list visibility or retention, and the one-month query window is not a retention guarantee.

For the GitHub implementation profile, a handler releases a hold through the pull-request comment /ceed resume HOLD-<id> <what you fixed>.
The Router MUST supply the hold ID in the check and alert, establish the commenter’s enrolled account identity, and require the explanation. The first valid command releases the hold under Section 4.3; later valid

commands receive a one-line already-released reply. A wrong, malformed, or unauthorized command receives a one-line explanation and leaves an open hold in place. Command syntax does not grant authorization or override an objection. GitHub notification preferences affect whether handlers see the alert; notification failure never releases the hold.

18.3 Source-control bindings

This subsection is non-normative. Implementers use this guidance to connect another source-control system to the same delivery principles. Version 1.0 defines the Git binding and GitHub profile used by the CEED Project implementation for Git and GitHub. Section 19 identifies implementation evidence still to gather.

1. Identify the change and exact candidate. Preserve the repository namespace, stable change and Change Passport identities, immutable base and head identities, and any constituent subset. Define the comparison being evaluated; changed inputs require a fresh decision (Sections 3, 7–8).

2. Supply the complete authoritative difference for that comparison, including deletions and both rename paths. Preserve the path semantics and repository-edge rules. Incomplete visibility, ambiguous comparisons, or unavailable required data fail closed (Sections 6–8).

3. Bind authority to exact inputs. Resolve trusted requirement and policy context, eligible actors, the route, grant, and full review predicate. Retain the precise reviewed proposal and evidence, declared control relationships, and continuing objection history (Section 4).

4. Establish issuance and inheritance. Verify the authorized proposal's relationship to the landing that changes only the Delivery Contract, identical bytes, and exactly the canonical path changed. Establish that the application candidate carries the issued version; keep authorized and landed identities distinct (Section 4.3).

5. Protect the control plane: paths, credentials, enrollment, policy and classification, gate configuration, and record writes. A producer cannot make a proposal authoritative by changing those controls; the human-administrator trust limit remains explicit (Sections 4 and 16).

6. Make decisions govern actual actions. Persist before approval, make every governed path wait, and preserve the checked candidate through landing or re-evaluate it. Establish bypass visibility and retention; an unobserved action cannot be claimed as approved (Sections 7, 11, 16, and 18.2).

7. Check the actual final build source and any combination. Preserve complete, disjoint constituent attribution and compatible release rules, or use separate landings. Hold deployment for required post-land checks (Sections 7–8).

8. Bind artifacts and deployment events. Preserve artifact digest and source provenance, distinct deployment identity, ordered checkpoints, exact evidence bindings, same-event strategy evidence, and truthful evidence basis. The deployment system waits for required decisions (Sections 9–10).

9. Preserve the continuing record: immutable Delivery Contract versions, authorization evidence, decisions, and production knowledge, with append order, consistent reads, idempotency, retention, and reporting snapshots. Revisions and retries cannot erase objections or authorize conflicting records (Sections 4.3 and 10–12).

Other Git hosts can retain the Git binding but need their own platform profile for review evidence, required checks, and bypass. Examples checked on October 18, 2026, are GitLab (GitLab Docs, Repository, https://docs.gitlab.com/user/project/repository/), Bitbucket Cloud (Atlassian, Create a Git repository, https://support.atlassian.com/bitbucket-cloud/docs/create-a-git-repository/), and Git in Azure Repos (Microsoft Learn, Collaborate on code, https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops). These sources identify Git hosting; they do not establish a CEED implementation profile.

A different source-control model needs its own explicit binding. Dated examples checked on October 18, 2026, are Scion, Anthropic's in-house source-control system (Anthropic Engineering,

Redesigning source control at Anthropic, published October 13 and updated October 15, 2026, https://www.anthropic.com/research/redesigning-source-control); Sapling, Meta's open-source source-control system (https://github.com/facebook/sapling); and Perforce P4, Perforce's version-control system (https://www.perforce.com/products/p4). Sapling also supports Git repositories (https://sapling-scm.com/); using a different client alone does not imply a different source-control binding. The binding follows the actual repository model and integration.

A builder using another source-control model first defines and tests its binding, then states any conformance claim against scoped evidence. Renamed SHA fields alone do not preserve the Git acquisition, authorization, ancestry, landing, and evidence requirements.

Identify the builder, platform, and Specification edition. Design alignment describes the principles implemented; a conformance claim identifies separately scoped evidence under Section 17. This lets reviewers distinguish a component result from a Full delivery implementation claim without inferring endorsement from a name.

19 Known open items

Reviewers use this list to distinguish settled requirements from implementation choices and evidence still to gather.

Implementation teams can use the open items below to plan the remaining evidence and design work. The list distinguishes specified requirements from unresolved choices in authorization, evidence, profiles, and reporting.

Issuance merges. Whether the CEED Project implementation performs issuance merges remains open. Exact authorization binding, the three post-land checks, and revert-and-re-propose recovery apply whichever component performs the merge.

GitHub bypass visibility and retention. Write-level visibility of bypass configuration and rule-suite retention remain empirical open items. The assessment results and their precedence are specified in Section 18.2. The GitHub App permission "Administration: read" was sufficient for the public rule-suite list and detail calls described there; the query window is not a retention guarantee.

Publication and test pairing. The publication date for the Conformance Pack Guide and the Conformance Pack is October 26, 2026. The Guide and Pack identify their own dates and exact Specification revision. Section 18 defines export layout and test coverage. Declaration validation and new security checks await implementation evidence; machine schema and profile versions have their own identifiers.

Strategy vocabulary and later adapters. Vocabulary granularity and later strategy adapters remain open. Version 1.0 has no strategy adapter; provider-record support requires empirical evidence.

Adapter conformance and assurance vocabulary. A distinct adapter profile and publication home for framework-defined assurance profiles remain open. Conformance addresses mechanics; assurance qualifies evidence strength for a named conformant scope.

Durable retrieval. The implementation still needs evidence that retained Contracts survive branch deletion and history rewrite. Section 11's requirement is settled; no new schema field is implied.

MCP scope and timing. Inclusion of an advisory Model Context Protocol (MCP) companion in the CEED Project implementation remains open. Its Contract awareness uses Section 6's path rules; protected Router gates enforce the issued authority.

Delivery-unit boundary. Version 1.0 still needs a definition of the delivery unit and its deduplication boundary counted by the Section 15.3 privacy thresholds.

Declaration validation schema. Version 1.0 still needs the filename and manifest registration name of the Enforcement Coverage Declaration validation-view schema.

Merge-check review decisions. Version 1.0 still needs the conditions under which a merge check returns review-required.

Authorization-block metric. Version 1.0 still needs a decision on whether invalid-path contributes to the authorization-block numerator.

Artifact-bound evidence reuse. Version 1.0 still needs a rule for carrying and validating the original deployment and checkpoint context when artifact-bound evidence is reused.

Appendix A Initial Change Passport-native metric definitions

Report implementers use these fixed cohorts and formulas so identical Passport snapshots produce comparable local measures. Appendix C names the local production-history metrics and defines their complete-snapshot scope label. Section 19 identifies the open invalid-path numerator decision.

Notation for reporting period p, evaluated as of generated_at: E is merge-check entries in p; C is distinct Change Passports whose first-ever merge-check occurred in p; C30 is the subset of C with 30 days elapsed since its first merge-check by generated_at; B is batch-admission-check entries in p; Eb is merge-check entries with batch_ref in p.

Each newly appended evaluation contributes to its applicable entry cohort; an idempotent return contributes no additional entry. Contract versions with identical content digests remain separate issued versions, and incident re-association changes effective state without adding a fresh production outcome.

Metric	Type	Formula	Interpretation
merge_evaluations_total	Count	|E|	Every recorded merge attempt in the period.
changes_evaluated_total	Count	|C|	One change enters the cohort when its first-ever merge-check occurs in the period.
authorization_block_rate_change	Rate	Changes in C whose first merge-check was blocked by out-of-allowed-scope, protected-scope, missing-evidence, policy-protection, unissued-contract-version, or invalid-contract / |C|	Late-overlap and unattributed-path are batch-integration failures and are excluded.
authorization_block_rate_attempt	Rate – local	Merge-check entries in E blocked by the authorization reason codes listed for authorization_block_rate_change / |E|	Attempt-level operational diagnostic.
scope_violation_rate_allowed	Rate	Changes in C whose first merge-check had at least one path finally classified out-of-allowed-scope / |C|	Exclusive path classification; protected paths are counted only in the protected rate.
scope_violation_rate_protected	Rate	Changes in C whose first merge-check had at least one path finally classified protected-scope / |C|	Exclusive path classification; out-of-allowed paths are counted only in the allowed-scope rate.

Metric	Type	Formula	Interpretation
evidence_missing_block_rate	Rate	Changes in C whose first merge-check failed missing-evidence / |C|	Does not assess evidence quality.
review_required_rate	Rate	Changes in C whose first merge-check was review-required / |C|	Measures recorded escalation.
protected_scope_touch_rate_30d	Rate	Changes in C30 with any protected-scope failure within 30 days of first merge-check / |C30|	Only changes with a full 30-day observation period.
retry_density_30d	Ratio	Merge-check entries for C30 within 30 days of first merge-check / |C30|	For non-empty C30, the value is at least 1; 1 means one recorded evaluation per change.
contract_versions_issued	Count	contract-issued entries timestamped in p	One change may issue several versions.
risk_tier_distribution	Distribution	Changes in C grouped by the risk tier on the contract-issued entry matching the Delivery Contract version used by the change's first merge-check / |C|	Context, not a score. The governing version is explicit.
batch_admissions_evaluated	Count – local	|B|	Candidates evaluated for batch admission or prospective-group compatibility before merge.
batch_admission_conflict_rate	Rate – local	B entries with conflict-detected / |B|	Depends on repository shape and queue configuration.
late_overlap_block_rate	Rate	Entries in Eb blocked on late-overlap / |Eb|	Integration-health measure; may be not_applicable when no batches exist.

Table 10. Metric definitions and cohorts. Fixed cohorts and denominators make recorded authorization and batch results repeatable. E: merge-check entries; C: changes first evaluated in the period; C30: C with 30 days elapsed; B: batch-admission entries; Eb: batch merge-check entries. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

Recorded production history – local only

The reporter may show distinct recorded deployment_ref values, initial recorded outcomes, rollback flags, automated outcome share, active late-incident associations, retractions, and recorded current classifications. Every label MUST include recorded or local-only. These figures MUST NOT be presented as complete deployment, gate-coverage, outcome-coverage, or incident rates until the later external denominators exist.

Appendix B Missing-status vocabulary

Use these values to explain absent data so readers can distinguish unknown, uncollected, unsupported, and withheld results.

Value	Meaning
not_collected	The configured process did not collect the field.
declined	The contributor explicitly declined optional context.
unknown	The source exists but the value cannot be established.
not_applicable	The measure does not apply, for example no batched merge activity.
not_observable	The required fact is outside the available Change Passport-native sources.
insufficient_sample	Publication privacy or sample-size threshold was not met.
invalid_source_data	The source violates a required invariant, such as duplicate non-identical initial outcomes.
unsupported_schema_version	The reporter cannot interpret the record version.

Table 11. Explaining missing data. Named absence values distinguish collection, observability, privacy, validity, and version limits. Source: CEED Specification and Change Passport Schema, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com.

Appendix C Normative clarifications

The references below identify each clarification’s governing rules.

Sections 4.1 and 6.1–6.2: schemas validate complete canonical Contract paths, identifiers, other repository paths, and scope patterns through their end boundaries. Paths and patterns exclude the specified Unicode controls; invalid policy patterns fail closed.

Section 6.1: changed repository paths can contain literal *, ?, [, or] when Git reports those names. Version 1.0 scope patterns exclude those characters except for the terminal /** subtree suffix.

Sections 6.1–6.2: every valid changed path, except the candidate’s own verified Delivery Contract path under Section 6.3, receives exactly one classification: authorized, out-of-allowed-scope, protected-scope, or policy-protection. Invalid-path and invalid-contract block evaluation; they are not path classifications. Equal-specificity allowed/protected matches resolve to protected-scope. Absolute repository-policy protection precedes candidate Contract scope-syntax evaluation.

Section 8.2: overlap takes precedence over unattributed-path.

Section 9.3: a deployment-check entry carries release_strategy, the authorized strategy from the Delivery Contract’s release_rule.strategy. Strategy verification compares the attested value directly, as text, with that release_strategy after any permitted versioned policy mapping. A declared boolean is not sufficient evidence of equality. Absent required strategy assertions block the checkpoint.

Sections 10.3–10.4: incident association state is keyed by the pair (deployment_ref, incident_ref) within one Change Passport. A retraction for another deployment cannot clear the association.

Section 11: an idempotent retry matches the existing logical decision tuple and byte-identical complete decision payload, excluding only the store-assigned entry number and applying the specified file_subset normalization. A matching tuple with different content is conflicting source data; no second entry is appended. New evaluations and incident events use new decision_id values; new authorizations use new

contract_version values. Retrying an older decision neither moves it later in append order nor makes it the latest effective event.

Sections 11–12.3: conflicting or otherwise invalid Passports are excluded from metric cohorts and counted in data_quality under Section 12.3.

Section 12 and Appendix A: local-only recorded production history deduplicates fresh production outcomes by deployment_ref. recorded_deployments_total counts distinct deployments with valid fresh production entries. recorded_initial_outcomes counts each once by initial status; recorded_current_classification counts each once after active late-incident associations. active_late_incident_associations counts active (passport_uuid, deployment_ref, incident_ref) associations. These measures cover the complete reporting snapshot, including entries outside the requested period. Report views MUST label this scope with the literal complete-snapshot.

Appendix A: risk-tier distribution uses denominator 0 and status not_applicable for an empty cohort.

Appendix B: missing-status tokens have the meanings defined there.