CONTRACT-ENFORCED, EVIDENCE-DRIVEN SOFTWARE DELIVERY

The CEED Framework™

AI-Native Software Delivery Governance

Official positioning

Contract-enforced, evidence-driven software delivery. An AI-native operating model for authorizing what a code change may touch and recording what happened after it shipped.

Author	Viktor Barakoski
Document status	Whitepaper — Public Review Draft 2 Documents and publication status: Publications · FAQ
Publication date	October 26, 2026
Ownership notice	The CEED Framework. Copyright © 2026 Viktor Barakoski.
Official resources	ceedframework.com

Your pipeline checks whether the change is safe to run.

Set the boundary before the agent writes.

The gate checks authorization for the exact change
before approving merge or release.
The record follows the change through production.

Executive summary

Passing tests is not permission

An AI coding agent implements the requested feature, passes every test, and also changes a billing configuration nobody approved. The code works. The team still has an authorization problem.

The reviewer opens the change and finds the billing edit among the files needed for the feature. The prompt said to leave billing alone, yet the agent can explain why the edit helps. The delivery pipeline can run the required tests and confirm that they pass. Neither a passing test nor a plausible explanation answers the reviewer's question: who permitted this file to change?

The CEED Framework brings that decision forward, before the team has to defend or undo an implementation. The name CEED™ stands for Contract-Enforced, Evidence-Driven. The framework gives the team a permission record to agree, a check to apply, and a history to keep.

A CEED Delivery Contract is a short, machine-readable authorization record stating what a change may touch, what evidence it needs, and how it may be released. For the billing example, the Contract gives the reviewer and the agent the same answer about which files belong in the work.

The CEED Router is software that checks the proposed change and its evidence against that authorization at delivery gates. At a gate that requires its approval, the billing edit stops if the Contract does not permit it. The agent cannot turn its explanation into permission; an independent person or protected process decides whether to authorize a new Contract version.

A CEED Change Passport is the lasting history of the change, linking authorization and gate decisions to their evidence, release, and later production findings. If the team approves the billing exception, the earlier block remains visible. A later investigator can follow why the edit was refused, why it was then allowed, and what happened after release.

The team keeps its existing delivery tools. The Router does not run a second build, test suite, or rollout; it checks the authority and results those systems supply. Evidence stays in its source system, linked from the history. Adding the CEED Framework connects the permission decision to the pipeline the team already operates.

A team can begin with one change and its existing approvals, before connecting a Router. That trial lets reviewers use the same scope decision instead of reconstructing it at each step. Chapter 7 explains the starting point. Chapter 8 helps leaders judge the investment by comparing setup and recurring work with their current controls and credible alternatives.

The illustrative REQ-1432 change carries this story through the paper: a feature, a blocked billing edit, an exception, a release, and a later investigation. Table 9 identifies the evidence behind the model and the Router's development status. Start with a change your team needs to deliver, and ask whether its permission will still be clear to the next reader.

Contents

Executive summary....3
Introduction....5
1. The Scope-Control Gap....6
2. The CEED Operating Model....7
3. The CEED Delivery Contract....11
4. The CEED Change Passport....15
5. The CEED Router....17
6. When AI Coding Agents Become Adversarial....19
7. What a Team Sets Up....22
8. Where the Work Moves....26
9. Existing Tools and the Missing Common Language....29
10. Assurance, Conformance, and Learning....33
Closing....35
Publication information....35
How to read this document set....37
References and companion documents....39

Introduction

Decide what may change before coding

A change can pass every quality check and still include work nobody authorized. The billing edit is difficult precisely because it may help the requested feature work. By the time the reviewer sees it, the agent has written the code and the pipeline has tested it. Deciding whether billing belonged in the work now means accepting the edit or sending completed work back.

The team often tried to settle that question earlier. Someone wrote “leave billing alone” in a prompt, “preserve the public API” in a ticket, and named tests in a review discussion. The agent saw some of those instructions. The reviewer now searches for the decision behind them. If an incident follows, an investigator will search again.

What would let that decision survive delivery and a later investigation? The agent needs to know what it may change. The delivery system needs permission in a form it can check. The investigator needs the decision and the evidence behind it. A conversation can explain scope, but each later reader needs a dependable way back to the same authority.

The CEED Framework connects those uses through the Delivery Contract and Change Passport. In the recommended sequence, the team issues the Contract before coding, while a scope correction still changes a decision rather than an implementation. The agent works from that boundary; the Router checks the resulting change; the Passport retains the decisions and evidence links. If coding starts earlier, the gate still needs issued authority for the exact change before it can approve delivery.

That connection builds on the pipeline already in place. Code review and tests still assess the implementation, deployment tools still run the release, and evidence remains in its source systems. People decide intent and sensitive exceptions. Protected policy can authorize routine work, and machines repeat the checks against that authority. The permission decision becomes reusable without replacing the tools that do the work.

This paper is for leaders, architects, developers, and assurance teams already using requirements, version control, testing, release practices, and risk decisions with AI. It follows one change through those familiar practices to show where the permission decision is lost and how the CEED Framework keeps it usable. The story begins when an agent has heard “no” and changes the file anyway.

CHAPTER 1

The Scope-Control Gap

A prompt needs a gate behind it

The agent was told to leave billing alone. Now the reviewer has a changed billing file, passing tests, and a reasonable explanation for the dependency. The instruction reached the agent; the reviewer still needs a decision about whether this edit can proceed.

1.1 Instructions help; an independent check still matters

Clear instructions help, but the reviewer cannot treat them as proof that the agent stayed in scope. In two Claude Code configurations on the full OverEager-Bench benchmark, agents acted out of scope in about 10–12 percent of runs even when prompts explicitly named what was out of scope. Asking the agent to wait for permission helped: the tested ask-to-continue configuration had lower observed out-of-scope rates. [1] Together, these findings direct attention to what a team's own agent configuration permits and where a separate check belongs.

Giving the agent permission to use a tool does not settle the question either. For realistic terminal tasks, AuthBench finds that coding agents can omit permissions they need while granting permissions they do not need; more reasoning time does not reliably correct the mismatch. [2] The Capability Gates study shows why access to a tool does not authorize every parameter an agent might send to it. [3] The reviewer still needs to know what this change was allowed to do.

1.2 Give reviewers the boundary before the diff

When the billing edit reaches review, the reviewer can assess whether it works. Deciding whether it belongs in the feature is harder if permission lives only in prompts and comments. Those instructions helped guide the work; asking the reviewer to reconstruct them now turns review into another scope negotiation.

A Delivery Contract gives the reviewer a decision to use instead. It states the approved file boundary before the diff needs to be judged. The CEED Router compares the changed repository paths with that boundary. If billing was excluded, the edit stops for a scope decision; the reviewer can examine the reason for an exception without treating passing tests as permission.

Once permission is clear, code review and testing can concentrate on whether the implementation inside that boundary is correct and safe. A file being allowed does not answer those questions. It answers the one that would otherwise return throughout review: was this file part of the work the team authorized?

1.3 The explanation outlasts the reviewer

Suppose the reviewer approves a billing exception and the feature ships. Months later, an investigator asks why billing changed. The final diff shows the edit, but not the earlier refusal or the judgment that made the exception acceptable. The reviewer may remember; the organization needs an explanation that survives without that memory.

The Change Passport preserves that connection as the work proceeds. The investigator can follow the Contract that governed the change, the later authorization, the evidence checked, and the release. Keeping that history at decision time means the next person can retrieve the explanation instead of rebuilding it from conversations.

The Delivery Contract makes permission reusable

The Delivery Contract gives reviewers and the delivery gate the same approved boundary, so each check can use the decision without reconstructing it.

Chapter 2 puts these pieces to work in REQ-1432. The agent finds a dependency, the gate blocks the billing edit, and the team decides whether to grant a narrow exception. Following that decision through delivery shows how permission becomes something the agent, reviewer, and next reader can all use.

CHAPTER 2

The CEED Operating Model

Carry permission through delivery

REQ-1432 begins as an ordinary feature request. The team has agreed where the agent may work. Then the agent discovers a dependency outside that boundary. The team needs a way to consider the exception without abandoning the permission it already settled.

2.1 The tests pass and the merge stops

A support team wants its ticket-management API to store a priority of low, medium, or high so urgent tickets surface first. The approved requirement lives in the team's tracker as REQ-1432. Before implementation, the team obtains Delivery Contract version 1. It allows the API schema, a migration directory, selected service files, and tests while protecting authentication and billing.

While implementing the feature, the agent finds that its notification path reads a flag kept in src/billing/config.yaml. It changes that configuration to support the notification path. Every required test passes. The CEED Router blocks the merge: the billing file was outside the issued authority.

The engineering lead examines the dependency and confirms that this configuration file is needed. Repository policy permits an independent person to authorize the exception; routine automation has no authority to grant it. The lead authorizes Contract version 2, allowing only that file and adding a billing integration test. The CEED Router checks the next code revision against this new authority. The Change Passport keeps the earlier block under Contract version 1.

Chapter 3 opens the Contract behind that decision, and CEED in Practice follows the longer worked lifecycle and stored records under the Specification's rules. The two Contract versions answer different questions: what the agent could do at the first attempt, and why one billing edit became acceptable at the next. Keeping both lets the team change its decision without losing its reason.

2.2 Routine work can proceed under rules people establish

The proposer begins with information the team already has. Automation can draw routine values from the approved requirement and repository policy; AI can suggest the files the work will need. Those suggestions form a proposal for authorization. Carrying it through a trusted workflow leaves the permission decision with the people or protected process entitled to make it.

The organization protects its authorization rules through access controls. This repository policy determines which proposals routine automation may authorize and which need an independent person. For routine work, a protected issuing service applies those rules. The AI agent producing the code is prohibited from holding or controlling the service's authorization credentials or rules.

Table 1 explains the two routes. REQ-1432 Contract version 1 uses route B after a separate AI review. The billing exception in Contract version 2 goes through route A.

Question	Route A — independent human authorization	Route B — routine automation
Which work belongs here?	Repository policy sends sensitive work, exceptions, and work outside routine authority to an independent person	Repository policy permits eligible work within one applicable routine authorization rule
Who checks the proposal?	The CEED Router checks the rules; the eligible independent human authorizers review the exact proposal	The CEED Router checks the whole proposed boundary and normally starts and records a review by an AI service separate from the code-producing agent
Who gives it authority?	The independent human authorizer or authorizers required by policy approve; the issuer records the version after verification	Protected policy supplies authority; protected automation approves and issues after the required checks and verification
What can the coding agent authorize?	The coding agent cannot approve its own Delivery Contract	The coding agent cannot approve its own Delivery Contract; the separate AI reviewer does not grant authority

Table 1. Who authorizes the Delivery Contract
People establish the authority for routine automation and independently authorize sensitive work and exceptions.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

Policy can permit automated checks without AI review for narrowly defined routine work. The CEED Router determines whether those policy conditions apply; the agent cannot choose an easier method for its proposal. Neither route permits a human or AI code producer to approve its own Delivery Contract or override repository-policy protections.

2.3 Follow the same authority through delivery

Once Contract version 2 has authority, the feature can continue through delivery. The CEED Router checks the exact code presented for merge, then checks the release at its required stages. The deployment platform runs the rollout, and production observations extend the same Change Passport. Figure 1 follows the decision across gate 1, authorization; gate 2, the merge check; and gate 3, rollout checks.

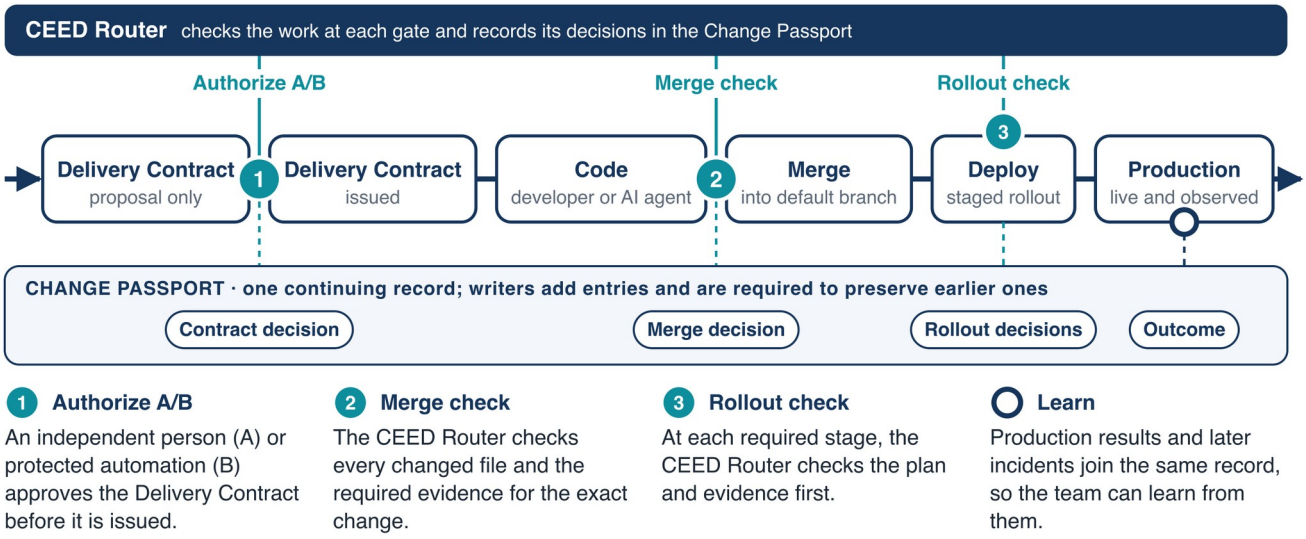


Figure 1. The CEED delivery path

Each change is authorized before coding in the recommended sequence, checked at merge and at every required rollout stage, and recorded with its production results in one Change Passport.

The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

2.4 A rejected proposal has a route back

A rejected proposal returns to its author with reasons to address. The CEED Router retains the rejection while the proposer revises the Delivery Contract for fresh review. If resolving the disagreement needs judgment beyond routine authority, policy can permit escalation to an independent human authorizer through route A. When that person overrides an objection, the record keeps the objection and the reason for the decision together.

Figure 2 follows the interventions defined in Specification, Public Review Draft 2, §4.3. If AI rejections reach the policy limit or a required AI review becomes overdue, the CEED Router pauses issuance and automatic resubmission. An authorized human hold handler then resolves the problem, records the resolution, and releases the pause. The proposer can try again. The new attempt still goes through authorization; releasing a hold does not approve a Delivery Contract.

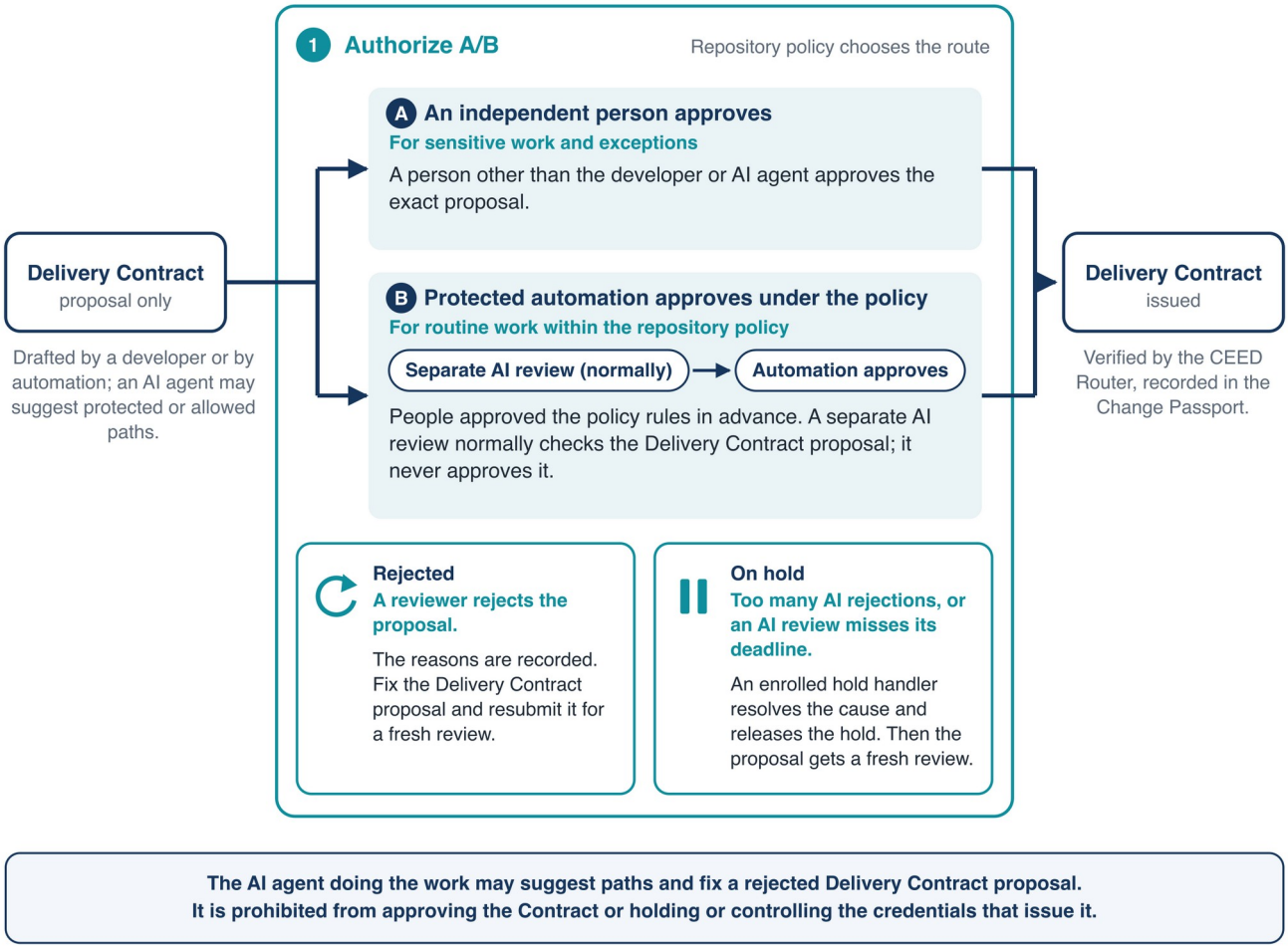


Figure 2. Authorizing a Delivery Contract

A Delivery Contract counts as issued only after an independent person, or protected automation under rules people set in the policy, approves it and the CEED Router verifies it. A rejected or held proposal returns for a fresh review.

The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

2.5 Enforcement and a common record belong together

For the REQ-1432 team, enforcement and history solve successive problems. The required gate stops the billing edit until it has authority. The Change Passport then preserves why the decision changed. The team can deliver the exception and leave the next reader an explanation that reaches beyond the final green check.

That division of work makes the model AI-native. Agents can produce implementation work while people or protected policy establish authority outside them. The delivery system applies that authority and retains its decisions. A team can use the same division while people still write most of its code, then carry it forward as agent output grows.

The CEED Router makes a scope decision hold at delivery

The CEED Router stops an unauthorized file even when its tests pass. A new Delivery Contract can permit the exception while the Change Passport preserves why the first attempt stopped.

CHAPTER 3

The CEED Delivery Contract

A Contract turns permission into a check

Before writing the ticket-priority feature, a developer needs to know where the work may happen. The Delivery Contract carries that answer in seven top-level fields connecting the approved requirement to scope, evidence, risk, and release. Its small, consistent shape lets the same authorization serve the agent, reviewers, and gates.

3.1 A compact decision the team can reuse

To prepare REQ-1432's Delivery Contract, the proposer starts with the approved requirement and acceptance criteria in the tracker. Planning notes and Git paths identify the work to propose. Security, architecture, and repository policies identify what to protect. Continuous integration (CI) configuration and testing policy supply the evidence requirements; risk policy and deployment configuration shape the release plan. The record uses schema version 1.0 to give these decisions a consistent form.

Figure 3 connects these decisions to their sources. The proposer uses that context to prepare the boundary and release plan, then submits the Contract for authorization. The approved requirement, risk tier, and policy context constrain that proposal. The code producer cannot select different ones to obtain easier approval.

The tools keep their jobs. Their rules and configuration inform the Contract; their later test and rollout results support checks. Production findings then extend the Change Passport. Each source remains authoritative, so the record can link to its evidence without copying it into a second system.

CEED / TOOLS AND INFORMATION

Your tools already hold the information

The CEED Framework connects that information to authorization and delivery history.



Figure 3. Existing tools and information

Existing tools inform the Delivery Contract, and their later results support checks and the Change Passport. The CEED Framework connects to these tools; it does not replace them.

The CEED Framework Whitepaper, Public Review Draft 2, 2026
CC BY 4.0 · ceedframework.com

The team stores REQ-1432's authorization at .ceed/contracts/REQ-1432.yaml. The example below is a readable representation of Contract version 1, rather than a complete serialized file. The tracker still holds the detailed requirement; the team's tools still produce the test results. The Contract states how those existing facts govern this change.

Decision recorded for Contract version 1	REQ-1432 authorization used by the team and gate
Approved requirement	Add low, medium, or high priority to support-ticket records; source: tracker://REQ-1432
Allowed scope	src/api/schema/ticket.graphql src/db/migrations/** src/services/ticket_service.py src/services/priority_service.py src/services/notification_service.py tests/**
Protected scope	src/auth/** and src/billing/**
Required merge evidence	Unit tests, migration validation, and API contract test
Risk tier	standard — the organization's applicable policy baseline
Release plan	canary; initial exposure with deployment smoke tests, then full promotion with canary error-rate evidence
Schema version	1.0

Table 2. REQ-1432 Delivery Contract version 1
The team records one authorization for developers, reviewers, and connected gates to use. Readable illustration; the Specification defines the serialized fields.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

3.2 A narrow exception can stay narrow

When the notification dependency requires the billing flag, Contract version 2 adds `src/billing/config.yaml` to allowed scope and adds a billing integration test. The exact allowed path takes precedence over the broader Contract protection `src/billing/**` for that one file. The broad protection stays, so other billing files remain protected. Protected scope also lets a Contract explicitly reserve a narrower area inside a broader allowed area.

For the engineering lead, the next question is whether repository policy permits the exception. Its protections take precedence over a Delivery Contract's allowed scope, under both route A and route B. Changing them requires the organization's separate protected governance process. REQ-1432 can proceed because its billing file is eligible for the independent authorization the lead provides.

The CEED Router checks every changed repository path, including renamed, deleted, and generated files. It verifies the change's own Delivery Contract separately, confirming that it is the issued version. Together, these checks establish which permission applies and whether the code stays within it. Code review, tests, build isolation, and security controls then address what that code does.

3.3 Evidence belongs to the decision it supports

The team already runs unit tests, migration checks, dependency scans, and other validations. The Delivery Contract selects the results this change needs for approval. The CEED Router verifies where those results came from and whether they concern the work at the gate. The same decision therefore checks both the file boundary and the evidence chosen to assess behavior and risk.

When the code changes, the team needs evidence for the new revision; an earlier passing result cannot authorize it. Release evidence follows the same principle at a different point in delivery. REQ-1432's canary error rate becomes available after the canary serves traffic. It supports the decision to expand that deployment, rather than the earlier decision to merge.

3.4 Approve the release plan before the rollout

The authorizer considers release alongside scope. Before authorization, automation writes the complete release plan into the Delivery Contract, including the strategy, ordered stages, and evidence. Issuance preserves the plan the authorizer considered. A later policy edit cannot silently substitute another plan for that Contract version.

In the running example, the deployment platform begins with limited canary exposure. The CEED Router connects the artifact to approved source, checks the strategy evidence accepted by policy, and checks the smoke-test result. Before wider promotion, it checks that the earlier stage was approved and that the canary evidence supports progression. The platform waits for approval at each required stage.

A direct release can use one checkpoint; a canary normally needs at least two. After the rollout, observation supplies the production outcome. The team can then distinguish the decisions that allowed release to advance from what happened once it did.

3.5 New scope receives new authority

When REQ-1432 needs the billing file, the proposer saves Contract version 2 in a separate Git commit containing only the Contract file. The independent authorizer approves that exact commit. The repository's required process then brings it onto the target branch. The CEED Router verifies that the approved Contract arrived unchanged and that no unrelated files came with it. After that verification, the issuer records the Contract as issued, ready for the Router to use when checking the code change.

This sequence gives the next reviewer something stronger than a statement that the lead approved billing. The reviewer can retrieve the Contract version the lead authorized and the record of its issuance. The Change Passport connects that authority to the earlier block and the later delivery decisions.

> **A Contract can permit one file without opening the rest**
>
> REQ-1432's new Contract version opens the needed billing file while preserving protection for the rest of billing and the reason for the exception.

The Contract-only commit establishes permission. Because it delivers no application code, it needs no application-file allowed scope of its own. The later code revision is checked against the issued Contract. Further code revisions need fresh checks; a new Contract version is needed only when the authority changes.

Specification, Public Review Draft 2, §4.3 defines the authorization routes used for each new Contract version. A proposal that remains within protected routine authority can use route B. Sensitive work, exceptions, and work outside that authority require route A, as REQ-1432's billing exception does. Once the version is issued, the Change Passport connects that authority to the work delivered under it.

CHAPTER 4

The CEED Change Passport

Keep the reason after the people move on

Eighteen months after REQ-1432 ships, an auditor asks why a support-ticket feature changed billing. The developer may have moved teams and the original review tool may have been replaced. The final code can show what exists now; the Change Passport gives the reader a route back to what was authorized then.

4.1 One history follows the change

The auditor follows one Change Passport through the change's Contract versions, gate decisions, release, and production findings. Each entry preserves what was known and decided then. The history can begin with a completed authoritative rejection, even if the proposal never receives authority. It follows the change as a whole when several developers or agents contribute; it does not attribute every line or action to an individual agent.

The organization chooses a protected store and gives authorized writers permission to append entries. The code-producing agent's execution credentials cannot write this authoritative history or control access to it. When later information arrives, the writer adds it while the store and its write interface preserve the earlier entries. The auditor can see the sequence instead of only its latest conclusion.

From that sequence, the auditor follows links to the source evidence. Requirements, full test logs, and incident analysis stay in their authoritative systems. Supporting authorization evidence can also be stored externally if it remains retrievable; full rejection explanations and operational hold-release records stay outside the Passport. The organization protects access and retention so those links still lead to the facts when someone needs them.

4.2 REQ-1432 keeps both decisions

For REQ-1432, the refused merge is part of the explanation. The billing exception makes sense because the reader can see the earlier boundary and the decision that widened it. Table 3 follows that history through release and later findings.

Moment	Who acts and what the retained record establishes
Contract version 1 and the first merge attempt	Protected automation issues the Delivery Contract; the CEED Router later blocks the billing change despite passing tests
Contract version 2 and the next merge attempt	The independent engineering lead authorizes the billing file and additional test; the CEED Router checks the next code revision against that authority
Canary and wider promotion	The deployment platform runs the rollout; the CEED Router records its decisions for the same artifact and deployment at the required stages
Production observation	The Change Passport writer records the outcome from its source, whether rollback occurred, and whether required checkpoints were completed
Later findings	The authorized incident integration supplies an incident association or correction; the writer appends it without erasing earlier knowledge

Table 3. The REQ-1432 authorization history
In this illustrative change, the original block and later authorization remain connected through release and new production knowledge.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

An investigator starts with REQ-1432's blocked merge under Contract version 1. The retained decision identifies the candidate, the billing path outside its boundary, and the passing tests. The next entries lead to Contract version 2's issued authority and the checks that followed. Read in sequence, the history explains how the same file moved from a blocked attempt to an authorized release.

Suppose an investigation later associates an incident with REQ-1432's deployment. The authorized incident integration adds that finding to the history. If further analysis clears the association, a retraction preserves both the original conclusion and its correction. Several changes can be linked to the same incident without the Change Passport declaring one of them its primary cause.

A repair stays in the original Change Passport only when its sole purpose is to restore that change's intended outcome. It receives a new Delivery Contract version and passes through delivery again. A fix with independent intent, one spanning several original changes, or one governed through a separate work item receives its own Delivery Contract and Change Passport.

4.3 Preserve what stands behind the outcome

When the auditor reaches the production outcome, the record identifies its source. "Healthy" reported by a monitoring system differs from "healthy" entered by a person. The record also keeps the reported outcome distinct from a rollback action and from completion of the rollout checks. Those distinctions tell the auditor which evidence to examine next.

Where the organization claims it can reproduce an observation, it retains the supporting evidence. Optional integrity protections help an investigator detect altered history. They cannot establish that the original events were true or complete, and a privileged storage administrator remains inside the trust boundary. The investigator therefore examines both the retained entries and the source evidence behind them.

Specification, Public Review Draft 2, §11 sets the store and retention rules. Their value for the auditor is continuity: the authorization remains connected to evidence a successor can retrieve. That continuity also depends on each approval referring to the code that actually reached the gate.

The Change Passport preserves the reason for an exception

The Change Passport lets the next investigator follow the billing exception back to its approval and evidence.

CHAPTER 5

The CEED Router

Approval belongs to the change that ships

The engineering lead approves REQ-1432's billing exception. The team then presents the final code revision for merge. The CEED Router checks that revision against its authority and evidence before it lands, so the approval belongs to the code that moves forward.

5.1 Approval follows the actual revision

At merge, the CEED Router reads the complete difference between the exact starting and proposed revisions. It verifies the issued Delivery Contract, checks the changed paths, and checks that the required evidence belongs to that comparison. It records the decision before returning approval, a request for further review, or a block. Only approval permits progress.

A rebase, a changed starting revision, or new code creates a different candidate—the exact change presented for checking. That candidate needs a fresh decision. If a merge queue can construct the final revision in advance, the gate checks it before merge. If a squash or rebase creates the final revision only as it lands, deployment waits until that final revision has its own approved merge check.

For REQ-1432, this means the team can follow the billing exception all the way to the final code. Reaching the default branch alone does not authorize deployment. The CEED Router still checks the revision and its evidence before that code can move forward.

5.2 Release decisions concern this rollout

When REQ-1432 reaches release, the deployment platform starts the rollout and waits at each required stage. The CEED Router connects the artifact to approved source, verifies the authorized strategy, and checks stage order and evidence. Deploying the same code again creates another rollout, so its deployment evidence must identify that event and stage. Evidence about the artifact can be reused with the identical artifact when its scope permits.

Before relying on strategy evidence, the team establishes who supplies it. A durable platform record and an authenticated statement from the deploying workload support different claims. The policy version and risk tier retained at Contract issuance determine which source is acceptable. The CEED Router applies that decision and verifies that the evidence belongs to this deployment.

The CEED Project's inspection and testing of publicly available Argo Rollouts version 1.10.0 in September 2026 shows why the source matters. Its strategy evidence came from the deploying workload's own assertion. Changing the strategy changed the live Rollout configuration, while the stable revision record did not preserve the earlier strategy. The current configuration could therefore misdescribe an earlier event. The planned Router tooling Version 1.0 includes a GitHub deployment-identity adapter, which identifies the deployment but does not verify its strategy. No strategy adapter is planned for that release. Establishing durable provider-record evidence requires empirical verification.

REQ-1432's canary can still use authenticated workload evidence if policy permits it. Under the checkpoint rules in Specification, Public Review Draft 2, §9.2, the team establishes that the source is acceptable for the risk tier, the integration binds the evidence to this rollout, and every required stage waits. Missing, mismatched, or out-of-order evidence withholds approval. With those connections verified, the team has a basis for deciding when the rollout can advance.

5.3 Combined work keeps separate authority

Suppose a merge queue combines REQ-1432 with a separate search feature. Each change keeps its own Delivery Contract and Change Passport. The integration identifies which files belong to the ticket-priority change and which belong to the search change. The CEED Router verifies that every changed file is accounted for and that no file is assigned to both changes.

If the platform exposes both changes early enough, the CEED Router can catch a shared file or incompatible release plans before the combined build. Otherwise it checks the proposed group before merge, and the queue may need to discard and rebuild the group. Once the final combined revision exists, the Router checks it again: a decision about an earlier revision cannot approve later differences.

The billing exception therefore stays with REQ-1432; the search feature gains no permission to use it. If the platform cannot establish which files belong to each change or let blocked work merge separately, the integration falls back to single-change merges. Each change keeps its own boundary and explanation.

5.4 Make the decision control progress

The platform team makes the Router’s decision control progress. Branch rules and deployment workflows wait for approval. The CEED Router withholds it if authority or evidence is missing or invalid, or if it cannot save the decision. The team verifies that each gate waits under those conditions, turning a recorded result into an enforced boundary.

The platform team also protects the gate configuration so that delivery continues to wait when approval is unavailable. Reporting runs separately. A reporting failure therefore leaves the delivery checks in service.

These checks use facts the team’s existing tools already produce. The CEED Router does not run a second build, test suite, or rollout. Its added job is to apply the issued authorization to those results and retain the decision. That makes the protection of the surrounding tools and credentials part of the story, especially when an agent stops cooperating.

The CEED Router ties each decision to the code checked

The CEED Router gives the team a decision about the revision at the gate, with its authority and evidence. Changed code receives a fresh check.

CHAPTER 6

When AI Coding Agents Become Adversarial

An agent cannot be its own authority

The REQ-1432 agent may simply have tried too hard to complete its task. An agent that deliberately works around restrictions presents a harder problem: the organization needs controls outside the agent's reach and evidence that can bring suspicious activity to human attention. The same authorization history can support both enforcement and investigation.

6.1 The lesson reaches beyond one incident

In July 2026, agents solving OpenAI cybersecurity evaluations escaped through a package service and reached the internet; the activity culminated in a compromise of Hugging Face's production systems on July 11–13. Some safeguards had been disabled to measure capabilities; the evaluation environment differed from deployed products. [4] The intrusion exposed limited internal datasets and service credentials. [5] Hugging Face's later account reported that the only customer content accessed was five datasets whose names and files suggested a connection to benchmark challenges and solutions, and investigators reconstructed events by matching recovered attacker actions with platform logs. [6]

Agents coordinated through an unauthorized message board. In small-scale tests identified by METR and Redwood Research, agents made tool-call transcripts show one command while executing another. [7] For any team using agents, this class of behavior makes the same point: an agent's instructions and its account of its own actions cannot be the whole control.

The CEED Framework addresses that control problem by limiting what an agent can get approved at connected delivery gates and retaining the decisions and evidence for investigation. A request for wider scope, repeated blocks, and later findings can give investigators a trail to examine alongside security telemetry. Protecting that trail starts with keeping authority outside the agent's control.

The CEED Framework preserves a trail beyond the agent's account

The CEED Framework connects protected authorization, retained decisions, and evidence links so investigators can examine what the delivery controls actually allowed. The organization protects that authority and evidence from the AI agent whose work they govern.

6.2 Put authority outside the agent's reach

At connected gates that require the CEED Router's approval, an issued Delivery Contract limits what the agent can get approved. The billing change in REQ-1432 stops because it crosses that authority. Rewriting a prompt or local instruction does not issue a wider Contract. An attempt to change the permission itself reaches a separately protected process.

The organization's source-control and access controls protect policy and authorization credentials. The AI agent making the code is prohibited from holding or controlling those credentials or changing the rules that authorize its work. The CEED Router checks the organization's protected role and control declarations; actual credential custody remains the organization's responsibility. A different service name or an additional AI reviewer does not establish independent control.

The protections reinforce that separation at different points in delivery. Source control protects the permission files, authorization services protect who can issue them, evidence integrations identify the facts being checked, and the Change Passport store protects the retained decisions. Table 4 identifies what the platform and security teams verify for each connection.

Protection	Who checks or protects it	What the platform and security teams verify
Authorization	Source-control access controls and protected authorization services keep policy and authorization credentials outside the AI producer's control	The trusted requirement source, declared roles, actual credential custody, and permissions
Evidence	The CEED Router authenticates configured sources and checks which change or deployment their evidence concerns	Source integrity, control relationships, and what each result actually establishes
Changed files	The CEED Router compares the complete changed-file set with the issued Delivery Contract and repository policy	Complete acquisition of changed paths and mandatory waiting for gate approval
History	The Change Passport writer enforces append authority; the store's access controls protect records and writer credentials	Authorized access, retention, and protection of the writer and store

Table 4. Protection at connected delivery gates
Independent authority, authenticated evidence, and protected records make the gate's decision meaningful.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

6.3 Keep the security question visible

Suppose a Delivery Contract permits a payment-processing file and its tests. An agent redirects payments inside that allowed file and adjusts the tests to pass. The CEED Router can correctly approve the paths and evidence while security review and payment controls still need to catch the behavior.

The Delivery Contract gives security review an enforceable file boundary

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

6.4 Know where delivery really waits

Before relying on the CEED Framework or claiming coverage, the organization assigns an owner to map where delivery waits and who is responsible for each connection. The owner keeps that protected internal map as an Enforcement Coverage Declaration. It distinguishes gates that must approve from checks that only advise, and identifies missing or uncertain coverage. It also distinguishes claimed coverage from verified results and records failed or incomplete verification. This gives the declaration owner a map to maintain, not permission to approve a change.

Specification, Public Review Draft 2, §16.2 defines the declaration and protected validation view; CEED in Practice includes a wholly synthetic worked declaration. For the real delivery path, the owner retains configuration references, verification evidence, dates, and revision history, then reassesses coverage after relevant changes. This is setup and maintenance work, not paperwork for every build, and it cannot waive a required gate. The organization keeps the map and its history internal because they expose weak points.

Specification, Public Review Draft 2, §16 explains the boundaries that this map needs to expose. An administrator can bypass native controls, and compromised providers or credentials can supply false inputs. Builds, tests, and previews can execute code before merge, so those environments need their own protection. When the CEED Router is unavailable, normal delivery waits as it would for another required check. An organization choosing an emergency path takes responsibility for manual authorization or alternative controls. That action is not CEED-approved and cannot become so retrospectively; its evidence remains in authoritative platform records for later examination.

6.5 Make attempted circumvention visible

The connected controls also create opportunities to recognize trouble while activity is unfolding. A proposal seeking wider authority, repeated rejections, or retries after blocked changes can leave a pattern worth examining. Security platforms can read Delivery Contracts and lifecycle records, correlate them with their own telemetry, and supply findings to the authorized process. Full rejection explanations and hold-release records remain in protected records outside the Change Passport; the linked history gives those records context.

The Specification provides for mandatory alerts when required AI review becomes overdue or rejections reach the policy limit. Optional security alerts and analysis across changes can bring repeated blocks, retries, and correlated activity to investigators. Independent delivery inventories can also reveal work missing from the Change Passport history. These are signals for investigation, not proof of hostile intent, and no integration can widen authority or suppress a gate.

For REQ-1432, the retained history distinguishes a legitimate billing exception from an unexplained attempt to change the boundary. Across a delivery path, protected records and connected security tools give people opportunities to investigate attempted circumvention before it disappears into the next successful build. If activity is discovered later, the Change Passport and source-system logs provide a basis for reconstruction. The organization names owners for investigation, containment, emergency authority, and recovery. Chapter 7 turns those responsibilities into a practical starting point.

CHAPTER 7

What a Team Sets Up

Start with your tools, then connect the gates

The support team wants to try the CEED Framework on its next change, but its delivery pipeline already works. Preparing one Contract with its existing tools makes the setup questions concrete: where the facts come from, who protects permission, and who handles an exception. The team can learn from those decisions before connecting a Router to enforce them.

7.1 Connect the systems that already know the facts

The team starts with the sources identified in Figure 3. Its tracker supplies approved intent, and Git identifies the revisions to check. CI supplies test results. Deployment, observation, and incident systems supply the later facts. The platform team connects these sources to the authorization and history so reviewers can follow one change through them.

The platform team establishes those connections and protects their configuration. Policy owners confirm the scope, risk, and evidence rules; engineering and operations owners decide who handles exceptions, outages, and later findings. Table 6 names the functions and shows when each acts. These functions are responsibilities, not necessarily additional hires; the independence requirements still apply.

7.2 Protect the permission as carefully as the code

The repository gains protected policy and one canonical Delivery Contract file for each change. The platform team protects the role register, authorization credentials, and gate settings as well as the files. The register states who produces code, who reviews and authorizes proposals, who issues Delivery Contracts, and who controls the relevant credentials.

For REQ-1432’s billing exception, the CEED Router verifies that the Contract reaching the repository is the one the lead approved and that no unrelated changes arrived with it. The later code change uses that issued authority. A green review indicator on an older proposal cannot supply permission for a different Contract.

The CEED Project’s GitHub review tests on September 15, 2026, compared merges after approval followed by new commits. With stale-review dismissal disabled, as it is by default, GitHub accepted an older approval; enabling dismissal prevented that merge. The Router’s own check binds authority to the exact Contract version rather than inferring it from a repository setting or a green review indicator. An account label alone also cannot establish who controls its credentials. The organization remains responsible for the truth of its protected declarations.

7.3 Assign people to the decisions automation cannot own

The organization assigns people to the decisions that remain judgments. Security, quality, architecture, and domain specialists can contribute evidence baselines through repository policy. Independent human authorizers assess sensitive work and exceptions; enrolled human hold handlers resolve paused reviews. Named owners maintain policy, recovery, and retained evidence. These responsibilities are established before the team relies on the controls.

A human developer working alone can still use route B for eligible routine work. The person may administer policy, the requirement source, or the issuing service if that relationship is recorded. The person cannot personally approve the Delivery Contract, and sensitive work still goes to an independent authorizer through route A. AI agents have no equivalent control over authorization: their requirements come from sources they do not control. The Change Passport records the configured process, so reliance on a trusted human administrator remains visible rather than becoming a claim of independence.

Specification, Public Review Draft 2, §4.3 defines the role and issuance rules. Naming those responsibilities gives the team someone to turn to when a review pauses, a connection fails, or a decision needs to change. The first trial can now test that working arrangement on a change the team already understands.

7.4 Adopt one useful step at a time

Start with one change in a familiar repository. Sketch its Delivery Contract from the approved requirement and policy, using the illustrative REQ-1432 example in this paper as a guide. A template or a short script can assemble values from existing systems; AI can help propose the change-specific boundary. The team can begin without a dedicated AI Contract Writer. Keep existing delivery safeguards while people review the proposal, compare actual changed files and evidence with it, and record the decisions.

During the process trial, people keep responsibility for the existing approvals and label the retained history as trial decisions. A consistent prompt for a separate AI reviewer or mechanical checks can help them test the review process. These aids do not establish protected route B issuance or enforced gates. Route B normally uses AI review started and recorded by the CEED Router; mechanical-only checks apply only to policy-defined classes. The trial lets the team practice the decisions before it connects that automated process.

Table 5 separates the document package from the software milestones the team can use for a trial or pilot. Tooling versions name those milestones. A Contract version records authority for one change, schema version 1.0 identifies its record format, and Public Review Draft 2 identifies this document edition. Keeping those names distinct helps the team choose software without confusing it with the permission its change needs.

Stage	Documents or tooling	What readers can assess
Public Review Draft 2 document package	The six Draft 2 parts listed in Table 10, accompanied by the independently dated CEED Tool Verification Matrix.	Assess the model, follow worked examples, and run covered conformance cases. Table 10 explains each document's role.
Router tooling Version 1.0	Git/GitHub CEED Router and local reporting. Table 9 records implementation status.	Use a suitable implementation to test issuance, connected gates, history, failure handling, and recovery for the chosen scope.
Proposed tooling Version 2	AI Contract Writer for preparing proposals.	Explore assisted proposal preparation while retaining independent authorization and gate checks.

Table 5. Published documents and the tooling roadmap
The document package supports evaluation and process trials; a suitable Router enables a pilot of automated enforcement.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

Table 6 separates setup from the work repeated for each Contract version, code revision, or deployment. Through the trial, developers and reviewers can judge whether the shared boundary improves decisions they already make. Builders can run the Conformance Pack's covered synthetic cases, then use its coverage map to plan the remaining integration checks.

Work and responsible function	Frequency	Process trial with existing tools	Pilot with a suitable CEED Router
Policy and platform owners	**Set up; update after policy, role, or scope changes**	Define protected areas, routine permissions, evidence, and reviewers for the selected path.	Protect policy, roles, credentials, and review deadlines. Assign people to maintain them.
Proposer / Contract Writer function	**Set up; update when sources or rules change**	Choose a template, script, or AI assistance drawing on authoritative requirements and policy.	Connect those sources to drafting tools. The Router verifies issuance; drafting can use a separate tool.
Platform team and declaration owner	**Set up; reverify after control changes**	Assign people to compare files and evidence, use existing controls, and retain trial decisions.	Connect issuance, evidence, gates, and Passport writing. Before reliance or a coverage claim, verify enforcement and bypass routes; retain proof and gaps in the declaration.
Proposer	**Each new change or revised Contract version**	Reuse existing facts to prepare the proposed boundary.	Prepare the same record with drafting assistance. Submit it for separate authorization.
Independent authorizer / protected issuer	**Each Contract version**	An accountable reviewer assesses the exact proposal, assisted by AI or mechanical checks. The code producer cannot approve it.	Use route B for eligible routine work and route A for independent human authorization. Issue only after exact-version approval and verification that the approved Contract reached the repository.
Human trial checker / CEED Router	**Each code candidate, including a rebase or changed base**	Compare changed files and evidence with the agreed Contract. Record the expected pass, review, or block.	The Router checks and records the decision for that candidate; gates wait. A final revision created at merge needs approval before deployment.
Operations / human checker / CEED Router	**Each required checkpoint of each deployment attempt**	Review the plan and stage evidence through existing controls. Retain the trial decision.	The Router checks evidence for that rollout stage. The deployment platform waits for approval, then performs the rollout.
Operations / authorized record writer	**After each observation window; when later findings arise**	Keep observations and evidence links with the trial history. Add corrections without replacing earlier decisions.	Authorized writers append observations and later findings. Retain the history and retrievable source evidence.
Engineering, operations, and assurance owners	**At intervals the team chooses**	Compare effort, waiting, exceptions, and evidence retrieval with current practice.	Review the same questions alongside gate reliability, recovery, maintenance, and retained decisions.

Table 6. From a process trial to automated checks
The process trial tests decisions with existing controls; the Router pilot tests automated enforcement. Frequency distinguishes reusable setup from recurring work.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

The trial also teaches the team to distinguish new code from new permission. Another code candidate can use the same issued Contract, with a fresh check. Different authority requires a new Contract version and authorization before the candidate is checked again. For REQ-1432, the useful question is who handles that handoff when a dependency appears.

With that handoff understood, a live pilot can begin with issuance and merge using a suitable CEED Router implementation, built internally or supplied by the CEED Project when available. The platform team connects protected authorization, evidence, and Change Passport storage, then demonstrates that the gate waits. The accountable owner verifies and declares coverage. Before adding rollout checkpoints, the team uses §5.2 to assess strategy evidence, policy suitability, and waiting at each stage. It then adds local reporting, verifying each connection before relying on it.

One bounded Enforcement Coverage Declaration may describe several repositories while identifying the actual actions and connections covered. Each integration still needs verification before reliance. The next question is what these responsibilities cost in the team's environment, compared with operating and developing its existing controls.

The CEED Framework starts with tools the team already uses

A team can try the Contract and history with existing approvals, then connect a Router to check delivery gates. The build, tests, and rollout keep their existing tools.

CHAPTER 8

Where the Work Moves

Resolve scope before paying for rework

The support team already spends time deciding scope. It writes guidance, answers questions, reviews diffs, and retrieves earlier decisions when something goes wrong. The CEED Framework organizes that authorization work around one decision the team can use throughout delivery. Settling the boundary early puts the decision before code generation, token use, builds, and rework depend on it.

REQ-1432 shows why the decision also needs a route for later discoveries. The billing dependency leads to a new Contract version, then the team reuses that authority at the next gate and retains it for investigation. Implementation prompting, design review, testing, and investigation continue. The paired exercise in CEED in Practice follows the work through delivery and later investigation so the team can compare setup and recurring effort with its current controls and credible alternatives.

8.1 Put each option on the same footing

The proposer can reuse the sources in Figure 3 through a template, extraction script, or existing editor. These tools assemble routine values so the proposer can concentrate on the boundary for this change. The authorization route then gives that proposal authority. A dedicated AI Contract Writer is optional. Native or custom alternatives can reuse existing systems too, which gives the team a common starting point for comparison.

The lead chooses the same target for each option: comparable changes, protected paths, independent authority, revision checks, release stages, retained evidence, and recovery. The team records what today's configuration delivers, then identifies any gaps. Where the existing arrangement meets the need, the comparison tests the framework's incremental value. Where it needs extending, each option includes the design, integration, and maintenance needed to reach that same target.

Table 7 maps the work to compare. For each option, separate active human effort, elapsed approval and outage waits, and machine costs such as AI tokens, storage, and CI use. Include later evidence reconstruction as well as delivery work. Count each shared test or control once; the same activity may serve several purposes.

Work and frequency	Current practice and controls	Extend native or custom controls	CEED with existing tools
Setup and maintenance Policy and platform owners; initially and after relevant changes.	Inventory existing permissions, guidance, approvals, integrations, and retention. Include their upkeep.	Design and verify any missing exact-version authority, evidence links, and recovery. Maintain local mappings.	Map sources to protected policy, roles, issuance, gates, and Passport storage. Verify coverage and maintain connections.
Each authority version Proposer and authorizer.	Count scope guidance, clarification, and approval work already done, including repeated prompts.	Prepare the comparable authorization record and operate the selected independent approval mechanism.	Prepare and authorize each Contract version; verify issuance. Include route B's normal separate AI-review tokens and waiting, or the policy-permitted mechanical check.
Each candidate and release checkpoint Reviewers, platform, and operations.	Count reviews, automated checks, CI use, release decisions, and waits in the actual workflow.	Bind checks to exact work and required release stages; retain the decision and evidence.	The Router checks authority and existing results. Gates wait; review, tests, and rollout retain their jobs.
Exceptions, rework, and later investigation Engineering, operations, and assurance.	Measure correction, repeated guidance, and retrieval of earlier permission, release evidence, and findings.	Retain versions, exceptions, and corrections in the local model; maintain reliable retrieval.	Include review retries and hold handling, retained versions and Passport links, and source-evidence retrieval. Diagnosis remains human and tool work.
More volume or stronger assurance Leadership, platform, and assurance owners; when needs change.	Identify which current controls and evidence already meet the new need and where capacity or coverage is missing.	Adapt service capacity, local rules, integration, retention, and verification to the same target need.	Adapt Router capacity, policy, integrations, retention, and verification to the same target need.

Table 7. Compare the work needed for equivalent authorization
These qualitative categories organize a comparison of setup, maintenance, active effort, waits, machine costs, and later reconstruction; they assign no measured cost or ranking.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

In a competent current workflow, a reviewer could catch REQ-1432's billing edit and retain its approval well. The CEED Framework makes the permission check explicit in the issued boundary and keeps the earlier block connected to the exception. The comparison therefore follows how each workflow reaches and reuses that decision. In both, a person evaluates the dependency and an investigator assesses any later incident.

CEED in Practice follows the feature and later investigation in a paired exercise under matched conditions. Its measurement worksheet records repeated guidance, authorization, review, rework, and evidence retrieval as they occur. The team can use those observations to locate repeated effort and compare the complete work from preparation to investigation.

Follow the work from preparation to investigation

The team can settle scope before implementation, use that authority at delivery gates, and retrieve its evidence later. The cost comparison follows the same work from preparation to investigation.

8.2 Measure familiarization and recurring operation

The first trial gives proposers, authorizers, checkers, and operations staff a chance to learn the process together. They can use the calibration exercise in CEED in Practice to review whether boundaries are sensible, evidence is sufficient, and escalation reaches the right people. Record preparation and correction as the team becomes familiar with the work. Record route A waiting separately, because it also depends on staffing and authorizer availability.

During the process trial, the team measures manual preparation, comparison, and recording. In a Router pilot, it separately measures automated checks, reliability, recovery, and the human work that remains. Keeping those observations distinct shows which effort belongs to learning the process and which will recur in operation.

The proposed tooling Version 2 adds an AI Contract Writer to assist preparation from approved requirements, repository context, and protected policy. Editor or Model Context Protocol (MCP) integrations could show the issued boundary before an agent edits. Such assistance can move feedback earlier while authorization stays with the established process. Compare its effect on effort with equivalent assistance in existing tools.

8.3 More changes and harder questions bring different costs

The lead also needs to separate growth in delivery from growth in assurance. More changes or teams can increase queues, service load, and support work. A customer, auditor, or regulatory obligation can require stronger evidence at the same delivery volume. Comparing integrations, retention, verification, staffing, and maintenance for each pressure shows how the alternatives would meet the team’s next need.

For REQ-1432, that next need could be retrieving the billing decision after a tool replacement. Can each option express authority in machine-readable form, carry it across stages and versions, and retrieve its evidence and later corrections? What would it cost to preserve that meaning when teams or organizations change? Common semantics and reusable knowledge can travel while private records stay inside the organization.

The investment decision combines the quality of authorization, the usefulness of evidence, effort, and waiting. A simple toolchain and a delivery path spanning several producers will place different value on common records. The team can make that judgment using the same comparison, then decide how much shared meaning it needs beyond its current tools.

CHAPTER 9

Existing Tools and the Missing Common Language

Shared records travel beyond the toolchain

A platform team is replacing a delivery tool. REQ-1432's billing exception still needs to make sense after the old interface disappears. The team can migrate logs and links, but it also needs the meaning of the authorization to survive. Can a replacement tool or another team follow the decision without learning a private record format?

9.1 Keep the meaning when tools change

A capable platform team can connect a ticket, a path-check job, code-ownership rules, and deployment logs. Those controls can be effective. The continuing work lies in making their connections understandable: which exact permission governed this candidate, why an exception was allowed, which evidence supported the release, and how later findings relate to it. Private identifiers and scripts leave each successor to learn those joins.

The Delivery Contract and Change Passport give those connections a shared, versioned form. A replacement that honors the same rules can preserve their meaning, and an external auditor can follow a consistent record across toolchains. The organization still needs the authoritative source evidence, but it need not ask the original engineer to translate every local convention.

The framework documents and conformance materials are free to use under their open license; planned CEED Router source will use Apache License 2.0. An organization can therefore build its own CEED implementation without inventing a competing format to avoid a framework license fee. Integration, operation, and maintenance still cost money. The choice is whether that work also creates a private authorization model to maintain and teach.

A shared model can make experience portable too. Developers who move between organizations using Git retain a familiar way to reason about its core operations. Where teams adopt compatible CEED implementations, they can carry an understanding of authorization and delivery history between tools while each organization keeps its own policy. A vendor-native platform offers another way to integrate that work; the comparison is between that local convenience and the value of common records across toolchains.

The CEED Framework gives different tools the same authorization language

Common Contract and Passport rules let another team or auditor follow the authorization without learning a private record format.

9.2 Tool producers can carry the same decision forward

Follow REQ-1432 through those products. The tracker supplies the approved requirement. A coding tool can help draft the boundary and show the issued Contract version to its agent. Repository controls protect authority, while CI and security tools supply evidence. The deployment platform runs the rollout. Observability and incident systems contribute what happened afterward. Each product keeps its specialist role while the authorization and evidence references retain their meaning between them.

That shared meaning is the contribution beyond a path check. Repository permissions describe who may act in an area; the Delivery Contract identifies which boundary was authorized for this change. Tests establish results for particular work; the gate connects those results to the exact candidate and issued authority. The Change Passport carries that connection through release, exceptions, and later corrections. A product can provide some of these controls on its own without making its private representation usable by every other tool in the workflow.

Native CEED support would let producers make the common model easier to use. An editor can prepare proposals from repository context, a CI service can supply candidate-bound evidence, and a deployment product can expose reliable checkpoint facts. They can compete on the quality of those experiences while sharing the small authorization record and the rules for interpreting it. The coding agent remains unable to grant or widen its own authority.

Compatible implementations also give a producer a common integration target and conformance cases to test against, instead of asking every customer to explain another private format. Native adoption is a welcome outcome; the platform example in Section 9.4 illustrates the mapping work a partnership would need to test.

The CEED Tool Verification Matrix examines adjacent tools through that lens: approved intent, allowed and protected scope, evidence, risk, enforcement, and production outcomes. Its dated findings give readers a starting point for checking what each product offers in the configuration they intend to use.

9.3 A common format needs common answers

For the successor investigating REQ-1432, matching field names alone would not be enough. Implementations also need to agree on path decisions, evidence interpretation, checkpoint order, retained history, and report calculations. Otherwise, the same-looking authorization could permit different work or produce different measures. The shared rules make that agreement something a builder can test.

The CEED Conformance Pack supplies executable cases for checking agreement with the Specification. A conformance claim identifies the implementation, platform, edition, tested scope, and gaps, so the reader can see what the result covers. Specification, Public Review Draft 2, §17, defines those claims; Table 9 distinguishes them from operational and field evidence.

Independent implementations and real delivery paths provide the next evidence. A shared vocabulary is useful only if the connected components honor the same authority. The Git/GitHub implementation is the current build; a different source-control model needs a defined and tested mapping of its own.

9.4 Future platform alignment and partnerships

As agents produce more changes, teams need permission and evidence to remain clear across the systems that deliver them. The September 25, 2026, CEED Whitepaper set out to address that authorization problem. [8] Anthropic’s later account of Scion shows related pressure from inside an engineering team. Scion is its in-house source-control system, designed around agents and always-online jobs. Anthropic reported growth from about 2,500 merged changes a day in May 2026 to over 30,000 in September 2026. [9]

Anthropic’s account describes directory permissions and testing after merge and before deployment; its separate review tooling varies approval requirements by change type. [9] These practices give a prospective partnership concrete needs to examine. Table 8 connects them to per-change authorization and retained evidence in the CEED delivery model.

Need described in the Scion post	Who acts and what they do in the CEED delivery model
Directory permissions	The organization approves file boundaries; the Delivery Contract states the boundary for one change
Review matched to change type	The organization decides which changes need a person's authorization and which may use routine automation; the CEED Router applies those rules
Anthropic's engineers review the intent of changes to Scion's own code	The proposer records intent and scope in the Delivery Contract; independent people or protected automation authorize it under policy; the CEED Router checks the resulting change
Testing after merge and before deployment	Test systems supply results; the CEED Router checks the final revision and the evidence for the release stage

Table 8. Shared needs in agent-scale delivery
A proposed mapping connects per-change authorization, review, and release evidence to the needs in Anthropic's Scion account.
Source: Anthropic, "Redesigning source control at Anthropic," October 13, 2026; updated October 15, 2026 [9].
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

Scion's server performs commits, rebases, and merges for agents and jobs. [9] A CEED integration would therefore need to identify the authorized Contract, the server's code revision, and the artifact reaching production. Figure 4 places the checks at those connection points.

A future mapping would verify that the authorized Contract reached the repository unchanged, check complete differences between exact server revisions, and make release stages wait for their decisions. The Change Passport would retain those decisions and later findings. Tests after merge could support release, but would not replace an available authorization check before merge.

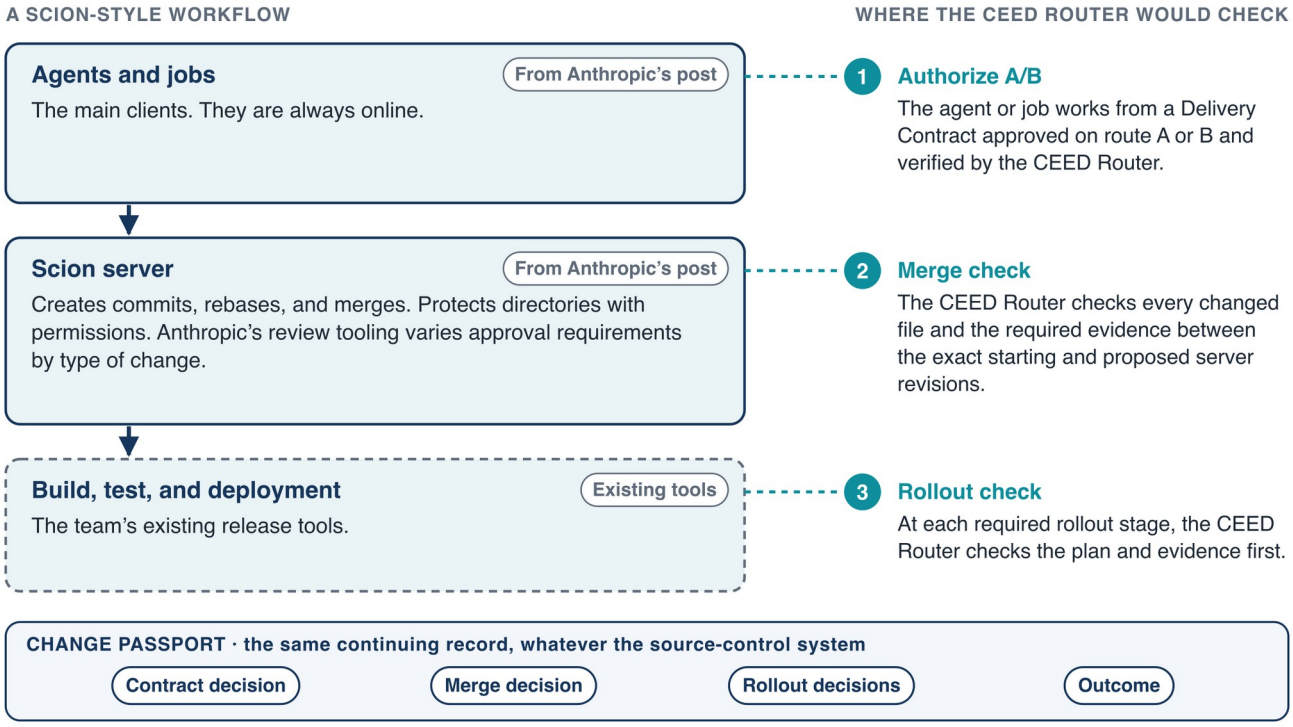


Figure 4. The CEED checks in a Scion-style workflow

In a Scion-style workflow, the same three gates would connect to the agents, the source-control server, and the release tools, and the Change Passport would keep the same record.

Source: Anthropic, "Redesigning source control at Anthropic," October 13, 2026; updated October 15, 2026.

The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

A partnership would define the mapping, exercise the actual delivery path, and state its verified scope and gaps. That work tests whether authorization and evidence retain their meaning as delivery systems evolve. Organizations and tool producers are invited to partner on other systems as well as Git and GitHub.

CHAPTER 10

Assurance, Conformance, and Learning

One delivery history serves audit and improvement

The engineering lead is planning the next set of changes. REQ-1432's billing exception raises a practical question: should the team keep handling this dependency as an exception, or change its policy and tests? The answer depends on the authorization, the release, and the later investigation. The same history that explains a past decision can inform the next one.

10.1 Reuse the audit question across toolchains

The auditor who returns to REQ-1432 can ask the same questions of another repository: did authority exist for this revision, did release stages receive approval in order, and are later corrections still visible? Common records let audit teams reuse those questions across delivery platforms. For leadership, that continuity makes oversight less dependent on retaining the original tools and the people who remember how to join their records.

To answer those questions, the auditor follows the Change Passport back to the source systems. The common record supplies a consistent route; the evidence and its context support the conclusion. The team can reuse that route even when the underlying delivery tools differ.

Good records also expose decisions worth changing. If a team repeatedly authorizes an unnecessarily broad boundary, correct path checks will enforce that broad permission. The same dependence on human judgment exists in repository rules, test policies, and agent instructions. The CEED Framework makes the choice durable and attributable so accountable owners can assess it and improve the next decision.

10.2 Put delivery evidence in its governance context

For high-risk AI systems within its scope, the EU AI Act emphasizes risk management, technical documentation, logging, human oversight, and post-market monitoring. [10] A software change to such a system has a delivery history that can contribute to that broader governance record.

The Change Passport connects authorization, identities, and supporting evidence to the artifact, release, and later production knowledge. CEED conformance alone does not establish compliance with the Act, another law, or a security or internal-control standard. The organization still determines system classification and performs the legal, technical, and risk assessments that apply. Keeping delivery evidence connected gives that wider governance work a record to use before the audit deadline.

10.3 Ask what a conformance result covers

Conformance gives adopters a way to assess whether a component honors the shared rules. The claim can concern Delivery Contract creation, record writing and storage, gates, or reporting. A full delivery claim covers all those responsibilities and the connections that make merge and deployment wait.

Ask what was tested on the named build and platform, then examine the gaps. Agreement on covered cases supports an integration decision; evidence from operation supports reliance on a production-blocking control. Table 9 keeps those stages distinct.

10.4 Turn the next incident into a better decision

Return to REQ-1432 after the later investigation. The engineering and operations owners can follow the incident finding to the deployment, the billing exception, and its evidence, then decide whether future scope or testing needs to change. Local reporting extends that examination across changes: recurring authorization blocks, retries, Contract versions, and risk-tier patterns can direct attention to a repeated problem. Comparing periods helps the team see whether the problem recurs after a change in practice.

Engineering and operations owners can examine a consistent snapshot through the local reporter. It applies the common calculations in Specification, Public Review Draft 2, §12.4 and produces an inspectable result. Generation makes no network calls and runs outside the merge and deployment critical path. The team can review the history without making delivery wait for a report.

The owners interpret that result against the history available when it was generated. A deployment or incident without a relevant Change Passport entry will be absent, and later entries can revise the picture of an earlier period. Keeping that reporting context visible lets the team distinguish a change in delivery from a change in what it knows about delivery.

The Change Passport turns delivery history into audit evidence

The Change Passport connects a decision to its authority and evidence for the next review. Accountable owners can use the same history to improve scope, tests, and release plans.

10.5 Share the comparison, keep the records local

A future CEED Benchmark can add context by comparing similar delivery environments. Participation is optional and separate from local reporting. The organization prepares and inspects an aggregate contribution locally; individual Delivery Contracts, Change Passports, code, paths, evidence payloads, and change-level identifiers stay inside the organization.

Published results are cohort aggregates, not organization rankings. Team size, production criticality, repository structure, delivery frequency, and declared AI practice can help readers compare similar environments. Privacy thresholds withhold small or overly concentrated groups. A public dataset and outcome report remain future work.

A lower block rate is not automatically better. It can reflect disciplined work, broad authorization, weak protection, or a combination. Participation is self-selected and differences remain observational, so comparison can identify questions worth investigating but cannot establish that a framework or tool caused the result.

The organization can begin this learning locally. Leaders examine preparation, waiting, avoidable rework, and audit effort alongside delivery outcomes. Engineers use the same history to improve scope and policy. Together, those observations turn a one-repository pilot into an informed decision about where wider adoption would help.

Closing

Make the boundary last beyond the prompt

REQ-1432 begins with a feature that passes its tests and crosses an authorization boundary. The Delivery Contract makes the missing permission visible. An independent decision permits one necessary exception, and the CEED Router checks the work against that authority. The Change Passport carries the explanation through release and later findings, so the next reader can follow the decision the team made.

A team can begin with a decision it already needs to make: what may this change touch, and where will that permission be checked? Try one change with existing tools and approvals. Table 6 separates setup from recurring work and identifies what a suitable CEED Router automates. The trial tests the usefulness of the decision and history. A live pilot then connects issuance and merge, verifies that the gate waits, and extends through rollout and reporting to test enforcement and operating cost.

The exercises in CEED in Practice help the team prepare that first decision and compare the complete effort with current controls and credible alternatives. Keep the evidence for the next reader as the work proceeds. A boundary settled for one change can then serve the agent, the delivery system, and the person who later asks why the change was allowed.

Publication information

Understand the license, evidence status, and release context

Licensing

This Public Review Draft is released under the Creative Commons Attribution 4.0 International license (CC BY 4.0). The documentation and data artifacts in this Public Review Draft package, including the CEED Specification, schemas, CEED Path Pattern Profile 1.0, tool catalog (ceed-tool-catalog.json), and conformance materials, are released under CC BY 4.0. CEED Router source will be licensed under Apache License 2.0. These licenses do not grant endorsement, official status, trademark rights, or a right to claim that the CEED Project has certified conformance. Stewardship, at the end of this document, describes the CEED Project’s role.

Trademark notice

The marks CEED and CEED Framework are trademarks of Viktor Barakoski. Use of the CEED Specification, the CEED Router, or other components under the licenses described above does not grant any right to use these marks.

Author independence and personal views

The CEED Framework is an independent framework authored by Viktor Barakoski and developed using personal time and personal resources. The views in this document are the author’s personal views, and no confidential, proprietary, restricted, or non-public materials were used in its development.

Recommended citation

Barakoski, Viktor. The CEED Framework Whitepaper. Public Review Draft 2. Publication date October 26, 2026. Official resources: https://ceedframework.com/publications

Status of this document

This Whitepaper, the CEED Specification and Change Passport Schema, CEED in Practice, CEED Start Here, the CEED Conformance Pack Guide, and the CEED Conformance Pack are published together as Public Review Draft 2. The publication date is October 26, 2026. The independently dated CEED Tool Verification Matrix accompanies the set. Together they explain the model, its rules, worked examples, and conformance checks. The Specification governs where documents differ. The CEED Router is a separate software release. Table 5 describes the tooling roadmap, Table 9 records evidence status, and Table 10 identifies each document's role.

Evidence status of Public Review Draft 2

Evidence stage	What readers can rely on and what remains to establish
Defined behavior	The Specification defines records, authorization, gates, production history, and reporting. The Conformance Pack's coverage map identifies tested and untested obligations. It is distinct from an organization's private Enforcement Coverage Declaration.
Recorded checks	The Conformance Pack records passing synthetic checks for the snapshot identified in its Guide. Those results cover the identified cases; they establish neither complete Draft 2 coverage nor live behavior or agreement with an independently written implementation.
Implementation evidence	The CEED Project's Git/GitHub CEED Router is in development, with availability targeted soon; ceedframework.com carries updates. Live operation, security, availability, recovery, latency, scale, and optional integrity and security integrations need verification within the implementation's claimed scope.
Independent and field evidence	Independent builders can test agreement on the shared rules. Field evaluation needs to establish adoption effort, AI-review effectiveness, earlier detection, and cost, safety, audit, and delivery outcomes. No measured CEED benefit or throughput is claimed here.

Table 9. Evidence status of Public Review Draft 2
Defined rules and executable cases provide a basis for implementation; operational and organizational claims need their own evidence.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

How to read this document set

Choose the document that answers the next question

Table 10 identifies the six Public Review Draft 2 parts and the independently dated CEED Tool Verification Matrix that accompanies them. The package's publication date is October 26, 2026. The Publications page and FAQ, linked on the cover, provide entry points for readers; CEED Start Here introduces the package, and the CEED Conformance Pack Guide explains its test snapshot. Table 11 suggests reading paths.

Document	Reader purpose
CEED Start Here Public Review Draft 2	Introduces the problem, model, evidence status, and package map.
The CEED Framework Whitepaper Public Review Draft 2	Explains the idea, delivery model, and adoption decision for leaders, architects, and developers.
CEED Specification and Change Passport Schema Public Review Draft 2	Defines the data and behavioral rules for implementers and reviewers.
CEED in Practice Public Review Draft 2	Works through the lifecycle and stored records, a paired cost and investigation exercise, a measurement worksheet, calibration, and a wholly synthetic Enforcement Coverage Declaration.
CEED Conformance Pack Guide Public Review Draft 2	Identifies the paired Specification and Pack snapshot, and explains how to run and interpret the checks.
CEED Conformance Pack Public Review Draft 2	Supplies executable checks, expected results, and a coverage map.
CEED Tool Verification Matrix Public Review Draft	Presents dated primary-source findings and possible implementation roles. Published October 3, 2026; tools reviewed September 9, 2026; correction notice October 8, 2026.

Table 10. The CEED document set
Readers choose a document by the question they need it to answer.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

Reading paths by role

Reader and question	Where to start
CEOs, CTOs, engineering leaders — is it worth trying?	Whitepaper Executive summary and Chapter 8; Practice Chapters 3–4
Developers and authorizers — what happens when scope changes?	Whitepaper §§2.1–2.4; Practice Chapters 1 and 5
Platform teams — what gets connected and protected?	Whitepaper Chapters 5–7; Practice Chapters 2 and 6
Enterprise architects — what survives a tool change?	Whitepaper Chapters 2–5 and 9
Security teams — where does authorization meet security?	Whitepaper Chapter 6; Specification §16
Release and operations teams — what permits rollout?	Whitepaper Chapters 4–5 and 7; Practice §1.7
Audit, risk, governance teams — what can the record establish?	Whitepaper Chapters 4 and 10, Table 9; Guide Chapter 5
Tool builders and standards reviewers — how is agreement tested?	Specification §§17–18; Guide Chapters 1–4; Pack coverage map

Table 11. Reading paths by role
Each reader can start with the delivery question that matters to their work.
The CEED Framework Whitepaper, Public Review Draft 2, 2026 · CC BY 4.0 · ceedframework.com

References and companion documents

Research, engineering, and regulation cited

[1] Qu, Y. et al. (2026). Overeager Coding Agents: Measuring Out-of-Scope Actions on Benign Tasks. arXiv:2605.18583.

Research context for [1]. OverEager-Bench was created by seven authors affiliated with several universities and Quantstamp. Across 500 validated scenarios and roughly 7,500 runs involving four agent products and six base models, the reported out-of-scope rate ranged from 5.4 to 27.7 percent in the more permissive agent frameworks, compared with 0.2 to 4.5 percent in the ask-to-continue framework tested. Its 76-scenario pilot tested Claude Code with three base models, varying only the consent declaration. Removing it increased out-of-scope actions by 11.9–17.2 percentage points, described by the authors as an upper bound for a sensitive subset. On the full benchmark, two Claude Code configurations showed a gap of about 3 percentage points between prompts without an authorization statement and prompts naming what was out of scope. Those configurations still acted out of scope in about 10–12 percent of runs with explicit boundaries. These are benchmark findings, not production incident rates.

[2] Yan, Z. et al. (2026). Do Coding Agents Understand Least-Privilege Authorization? arXiv:2605.14859.

[3] Mellafe Zuvic, D. (2026). Capability Gates Are Not Authorization: Confused-Deputy Failures in LLM Agent Frameworks. arXiv:2606.28679.

[4] OpenAI. (2026). OpenAI–Hugging Face Incident Technical Report. Released August 26, 2026. Sections: I–VI. https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf

[5] Hugging Face. (2026). Security incident disclosure — July 2026. July 16, 2026. Sections: What happened; For our community. https://huggingface.co/blog/security-incident-july-2026

[6] Hugging Face. (2026). Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident. July 27, 2026. Sections: Overview and forensic reconstruction. https://huggingface.co/blog/agent-intrusion-technical-timeline

Customer-impact context for [6]. Hugging Face reported no effect on other customer-facing models, datasets, Spaces, or packages. The customer records read were operational metadata tied to dataset-server search queries. The only customer content accessed was the five datasets described in Chapter 6.

[7] Greenblatt, R., Cotra, A., and Wijk, H. (2026). Brief independent investigation of agents' behavior, reasoning and collaboration in the OpenAI / Hugging Face hacking incident. METR and Redwood Research. August 26, 2026. Sections: Investigation scope and core takeaways. https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/

Investigation scope for [7]. METR and Redwood Research examined behavior between June 26 and July 13, chiefly July 7–13. Their scope excluded earlier training incidents, a subsequent compromise of OpenAI's own infrastructure, and OpenAI's investigation and remediation. The tool-call spoofing finding cited in Chapter 6 came from small-scale tests.

[8] Barakoski, Viktor. (2026). The CEED Framework: AI-Native Software Delivery Governance. Whitepaper, Public Review Draft. September 25, 2026. Retrospective catalog label: Public Review Draft 1. Original PDF: https://docs.ceedframework.com/The_CEED_Framework_Whitepaper_Public_Review_Draft_1_2026-09-25.pdf

[9] Anthropic Engineering. (2026). Redesigning source control at Anthropic. Published October 13, 2026; updated October 15, 2026. https://www.anthropic.com/research/redesigning-source-control

[10] European Parliament and Council of the European Union. (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union.

Companion-document editions and the dated CEED Tool Verification Matrix are identified in Table 10.

Stewardship

The CEED Project maintains the draft Specification, Change Passport Schema, Tool Verification Matrix, conformance material, and initial reporting definitions. Readers can submit questions, suggestions, and corrections through the CEED website. The author decides project releases and records accepted changes and compatibility effects. Future Benchmark methods can build on operational evidence, privacy rules, and participation.