

Round 22

When Does an Audit Expire?

Dependency-Triggered Revalidation for Rapidly Changing Agent Systems

A claim-lifecycle protocol, actual repository change audit,
and fixed-budget experimental plan

13 October 2026

Research paper / CTAG-0.1 proposal / D22-A deterministic diagnostic

No model-quality benchmark, provider experiment, security evaluation, or production deployment was run.



Contents

This publication separates frozen observations, current-state claims, external findings, and Encevra hypotheses.

Evidence status at a glance	5
Abstract	5
1. Archive check and carry-forward	6
1.1 Numbering	6
1.2 What Round 21 established	6
1.3 Relevant earlier rounds	6
1.4 Why the Round 21 handoff is not treated as completed	7
2. Research question, novelty, and decision rule	7
2.1 Research question	7
2.2 Novelty relative to Rounds 9, 10, 20, and 21	7
2.3 Success criterion	7
3. Scope and exclusions	7
In scope	7
Out of scope	8
Unit of analysis	8
4. Definitions	8
4.1 Historical claim	8
4.2 Current claim	8
4.3 Stale claim	8
4.4 Contradicted claim	8
4.5 Revalidation	8
4.6 Dependency	9
4.7 Safe selective revalidation	9
5. The observed trigger: Encevra changed while the assurance claim stayed still	9
5.1 Actual change set	9
5.2 What changed materially	10
5.3 What did not change enough	10
5.4 The key logical distinction	10
6. Literature map and competing positions	11
6.1 Provenance-first position	11
6.2 Assurance-case position	11
6.3 Retest-all position	11
6.4 Selective-regression position	11
6.5 Continuous-assurance position	12

6.6 ML-systems position	12
6.7 Brain-inspired analogy	12
6.8 Synthesis	12
7. Claim and evidence table	13
8. Proposed mechanism: Change-Triggered Assurance Graph (CTAG)	13
8.1 Claim card	13
8.2 Claim states	14
8.3 Dependency types	14
8.4 Change event	14
8.5 Expiry algorithm	15
8.6 Why negative claims need wider dependencies	15
8.7 External contracts	15
8.8 Release rule	15
8.9 Integrity and independent review	16
9. Worked example: a new provider and custom engine path	16
9.1 Before the change	16
9.2 Change event	16
9.3 Naive outcomes	16
9.4 CTAG outcome	16
9.5 Revalidation	17
9.6 What this example does not establish	17
10. D22-A: deterministic code-change diagnostic	17
10.1 Purpose	17
10.2 Inputs	17
10.3 Baselines	17
10.4 Execution	18
10.5 Actual-diff results	18
10.6 Constructed mutation-suite results	18
10.7 State-transition result	18
10.8 Interpretation	18
11. Competing explanations and alternatives	19
11.1 “This is just ordinary regression testing”	19
11.2 “A signed provenance graph is enough”	19
11.3 “Just timestamp every claim and expire it after 30 days”	19
11.4 “Retest everything after every commit”	19
11.5 “Human reviewers can remember what matters”	19
11.6 “The five Round 21 claims happened to remain true, so expiry was unnecessary”	19
11.7 “The result is circular”	19

12. Falsifiable hypotheses	20
H22-1: stale-claim containment	20
H22-2: selectivity	20
H22-3: negative-claim protection	20
H22-4: historical integrity	20
H22-5: mismatch-triggered efficiency	20
H22-6: independent derivation value	20
13. Proposed experiment E22-B	20
13.1 Objective	20
13.2 Preregistration	21
13.3 Evaluation stream	21
13.4 Independent gold set	21
13.5 Arms	21
13.6 Controlled total compute	22
13.7 Primary metrics	22
13.8 Secondary and regression metrics	22
13.9 Ablations	22
13.10 Statistical analysis	23
13.11 Stop conditions	23
14. Expected failure modes	23
14.1 Missing dependency edge	23
14.2 Over-broad dependency tag	23
14.3 Semantic change hidden in an unchanged path	23
14.4 Generated-code noise	23
14.5 Claim wording drift	24
14.6 Revalidation procedure drift	24
14.7 Selector-author leakage	24
14.8 Current-state availability collapse	24
14.9 False precision	24
14.10 Privacy and intellectual-property leakage	24
15. Limitations	24
16. Implementation sketch without production claim	25
17. Round 23 handoff	25
18. Reproducibility and artifact inventory	25
19. Sources	26
20. Final claim boundary	27

Evidence status at a glance

This round distinguishes four epistemic categories throughout:

- **Established for the frozen Enceavra commits:** directly inspected code, archive text, or deterministic diagnostic output.
- **Established in cited external work:** a result or specification reported by a primary paper, standards body, or official technical source. Transfer to Enceavra may still be limited.
- **Mixed or conditional:** supported only under stated assumptions, with important counterexamples or domain limits.
- **Enceavra hypothesis:** proposed here and not yet established by a prospective production experiment.

The central result is narrow but consequential. The Round 21 diagnostic remains historically valid for its frozen commit. It does not automatically remain a description of the current product. Between the Round 21 commit and the current inspected commit, 127 files changed, including new provider, model-routing, custom-engine, queue, evidence, API, and release paths. A manually audited set of 17 changed files can affect the scope of Round 21's current-state assurance claims. The existing measurement collector itself did not change and still does not implement the attempt-bound, atomic, fail-closed gate proposed in Round 21.

Round 22 proposes the **Change-Triggered Assurance Graph (CTAG)**: a typed graph that binds each current claim to the code, configuration, external contracts, tests, and evidence on which its validity depends. Relevant changes move a claim to `stale_needs_revalidation`; historical results remain frozen; unknown execution paths fail closed. CTAG is a proposal, not a production feature.

Abstract

Reliable AI systems need more than evidence attached to an answer. They also need a disciplined way to know when yesterday's evidence no longer supports today's claim. This problem becomes acute in multi-agent systems because the relevant system is larger than a single model: providers, prompts, serialization, routing, fallbacks, tools, evidence transforms, storage, release logic, and external contracts can all change independently.

Round 21 showed that Enceavra's measurement path lacked a complete attempt-bound receipt and proposed a fail-closed measurement gate. After that round, Enceavra changed substantially. The historical Round 21 result remains true of the commit it measured, but statements about what the current system does require revalidation. This round asks a distinct question: **Can a typed dependency graph preserve immutable historical findings while selectively expiring and revalidating only the current claims that a system change could invalidate?**

The literature supplies pieces but not a complete answer. W3C PROV and SLSA represent lineage. SACM represents explicit claims, context, arguments, and evidence. Safe regression-test selection formalizes conservative subset selection under explicit assumptions. Dynamic-assurance research shows that affected assurance subgraphs can be recomputed. ML-systems work warns that dependencies extend beyond code. None of these sources alone determines whether a specific claim about a rapidly changing agent system is still current.

Round 22 therefore proposes CTAG, a claim-level validity protocol with typed dependencies, explicit validity predicates, negative-claim coverage rules, external-contract versions, immutable historical states, and fail-closed handling of unknown paths. A deterministic diagnostic, D22-A, was run on the actual post-Round-21 diff and a constructed 32-case mutation suite. On the actual diff, a measurement-file watchlist selected 0 of 17 manually audited critical paths; a filename heuristic selected 12 of 17 plus 25 false positives; retest-all selected all 17 plus 110 false positives; and the manually authored CTAG manifest selected the 17 audited paths. On the constructed suite, CTAG exactly routed all 32 cases. These are routing checks against

manually authored labels, not evidence that CTAG can discover dependencies automatically or improve production reliability.

The next step is a prospective shadow implementation that records change events, marks claims stale before release, and measures stale-claim escape rate, false invalidation, time to revalidate, and fixed-budget verification yield across real commits. Until that experiment is run, CTAG remains a falsifiable architecture hypothesis.

1. Archive check and carry-forward

1.1 Numbering

The Encevera Research overview, rounds index, publications list, registry, and full manuscripts for Rounds 21, 20, 10, and 9 were inspected at commit [9441b602f7b65528734bd46c5af3a108b3a01a0d](#) [S01]. The archive ends at Round 21. No later completed round was found. This paper is therefore Round 22.

1.2 What Round 21 established

Round 21, **Attempt-Bound Measurement: A Fail-Closed Gate for Auditable Agent Runs**, audited the then-current measurement path. Its frozen results include:

- 34 of 34 deterministic checks passed in D21-A.
- Of 31 required audit fields, 10 were present, 9 partial, and 12 absent.
- The inspected path did not commit a complete provider request and response envelope.
- Provider terminal status was not fully recorded.
- Measurement rows were not committed atomically with the terminal run state.
- Release was not gated on a complete attempt-bound receipt.

Round 21 did not run a model-quality or security benchmark. It proposed, rather than validated, an attempt-bound measurement gate. Its handoff called for the smallest real shadow implementation before stronger claims or the deferred Round 18 repair pilot resumed [S01].

Those statements contain two different kinds of claim:

- 1 **Historical claims:** what D21-A observed at a named commit. These remain true unless the artifact or report itself is corrupted.
- 2 **Current-state claims:** what the live codebase presently records or fails to record. These can expire when relevant code, configuration, data, or external contracts change.

Round 21 did not define a machine-checkable rule for separating these lifecycles.

1.3 Relevant earlier rounds

- Round 20 developed measurement disclosure and independent witnessing. It emphasized receipts and the distinction between an observed outcome and a privileged rewrite.
- Round 10 developed evidence exchange and selective revision. It asked how one agent should expose reasons and evidence so another can revise the right parts of an answer.
- Round 9 studied reusable procedures learned from verified experience.

Round 22 does not repeat those questions. It asks when the claims produced by any of those mechanisms should stop being presented as current after the underlying system changes.

1.4 Why the Round 21 handoff is not treated as completed

The current code still lacks the complete attempt-bound gate. This round does not claim otherwise, does not run the deferred effectiveness pilot, and does not upgrade the Round 21 hypotheses. Instead, it addresses a newly exposed prerequisite: the system changed across multiple execution surfaces before the proposed gate was implemented. A research program that cannot distinguish frozen evidence from current assurance risks silently carrying old conclusions forward.

2. Research question, novelty, and decision rule

2.1 Research question

Can a typed claim-dependency graph automatically mark only the assurance claims affected by code, configuration, data, or external-contract changes as stale, preserve historically valid results, and require the right bounded revalidation before any claim is represented as current?

The question is consequential because both obvious alternatives are poor:

- Treat every old finding as current, and stale claims escape.
- Revalidate every claim after every change, and assurance becomes too expensive, slow, and noisy to sustain.

It is falsifiable because a proposed graph can be tested against independently labeled changes. It fails if it misses an affected claim, expires too many unaffected claims, cannot contain unknown paths, or consumes so much revalidation budget that a simpler policy performs better.

2.2 Novelty relative to Rounds 9, 10, 20, and 21

Prior work	Its main object	Round 22's distinct object
Round 9	Reusable procedures learned from verified experience	Validity of a procedure-derived claim after dependencies change
Round 10	Evidence exchange and selective answer revision	Expiration and revalidation of assurance claims across system versions
Round 20	Measurement disclosure and independent witnessing	Whether the evidence-to-claim link is still current at a new commit
Round 21	What a provider-bound attempt receipt must capture	When claims about that capture path must be rechecked

Round 22 is not another evidence packet, confidence score, memory store, or verification-budget allocator. It is a lifecycle protocol for claims.

2.3 Success criterion

CTAG is useful only if, under fixed total verification budget, it reduces stale-claim escape without producing an unacceptable rate of false invalidation or current-state unavailability. A perfect score on authored fixtures is insufficient. The decisive test is prospective performance on previously unseen real changes with an independently constructed impact gold set.

3. Scope and exclusions

In scope

- Claims about agent execution, evidence handling, measurement, release, and current implementation state.
- Code, configuration, model routing, external provider contracts, schemas, and evidence semantics.

- Historical versus current claim status.
- Negative claims such as “the system does not capture field X.”
- Selective revalidation under fixed compute and time budgets.
- Public-copy rules that prevent stale claims from appearing current.

Out of scope

- Proving that Encevera answers are correct.
- Evaluating model intelligence, calibration, helpfulness, or safety.
- Re-running the Round 18 repair pilot.
- Claiming that ABMG or CTAG is deployed.
- Automatically inferring a complete dependency graph from arbitrary source code.
- Reproducing proprietary prompts, routing thresholds, memory policies, or customer data.
- Claiming that neural memory and software assurance are the same mechanism.

Unit of analysis

The unit is a **claim card**, not a document and not a model response. One paper can contain historical facts, current facts, interpretations, and hypotheses with different validity lifecycles.

4. Definitions

4.1 Historical claim

A proposition explicitly bound to a frozen system commit, evidence commitment, procedure version, and observation date. New code does not make the old observation false. It may make it irrelevant to the current product.

4.2 Current claim

A proposition intended to describe the presently designated system or release candidate. Its validity predicate includes a system version and a set of dependencies that must remain compatible with its evidence.

4.3 Stale claim

A current claim whose validity predicate can no longer be established because at least one declared dependency changed, an external contract exceeded its inspection window, or an unknown execution path entered scope. “Stale” means “requires revalidation,” not “false.”

4.4 Contradicted claim

A claim for which new valid evidence supports its negation or makes its scope impossible. Contradiction is stronger than staleness.

4.5 Revalidation

Execution of the claim's named procedure against the candidate system and required external contracts, producing a new evidence commitment. Re-reading an old report is not revalidation.

4.6 Dependency

Any artifact, interface, semantic rule, or external condition whose change could make the claim's evidence insufficient. Dependencies may include code, schemas, configuration, model/provider contracts, data transforms, tool versions, release paths, and operational assumptions.

4.7 Safe selective revalidation

A selector is safe relative to a declared universe and assumptions if it omits no claim that the change could invalidate. As regression-test research emphasizes, “safe” is conditional, not universal [S07].

5. The observed trigger: Enceavra changed while the assurance claim stayed still

5.1 Actual change set

The exact Lovable diff from the Round 21 commit `311898162201788e8afa4bf5202a36e80f86ffd0` to the current inspected commit `9441b602f7b65528734bd46c5af3a108b3a01a0d` contains:

- **127 changed paths**
- **83 new paths**
- **44 edited paths**

The full list is in `observed_change_set.json` [S02]. The change set added or altered provider integration, routing, custom engines, engine versions, public API behavior, queue execution, cancellation, evidence labeling, deployment policy, cost accounting, schemas, and release-adjacent code.

Seventeen changed paths were manually classified as capable of affecting at least one Round 21 current-state assurance claim:

- 1 `drizzle/migrations/0004_engine_studio.sql`
- 2 `drizzle/migrations/0005_consumer_engine_limits.sql`
- 3 `src/lib/ai-providers.server.ts`
- 4 `src/lib/default-roster.ts`
- 5 `src/lib/deployment-policy.ts`
- 6 `src/lib/model-pricing.ts`
- 7 `src/lib/panel-core.server.ts`
- 8 `src/lib/engine-profiles.ts`
- 9 `src/lib/engine-run.server.ts`
- 10 `src/lib/engine-registry.ts`
- 11 `src/lib/engine-studio.ts`
- 12 `src/lib/run-queue.server.ts`
- 13 `src/lib/evidence-audit.server.ts`
- 14 `src/lib/verification-state.ts`
- 15 `src/lib/evidence-limits.ts`
- 16 `src/routes/api/public/v1/panel.ts`

17 `src/routes/api/public/v1/runs/$id.ts`

This is a manual audit judgment, not an automatically discovered ground truth.

5.2 What changed materially

Direct inspection found these relevant expansions [S02]:

- A new provider path and provider-specific serialization behavior.
- New reasoning configuration and provider policy checks.
- Custom engine profiles and versioned engine execution.
- New queue and public API paths that can select engines and attach end-user context.
- A cancellation path that writes a terminal state.
- New evidence verification labels and evidence-size controls.
- New schema and purchase/quota paths that change the execution surface.

These changes do not prove a regression. They do invalidate the assumption that the Round 21 audit covered every current path.

5.3 What did not change enough

The core measurement files inspected in Round 21 were unchanged in the diff:

- `src/lib/run-measurement.ts`
- `src/lib/run-measurement.server.ts`
- `src/lib/panel-runs.server.ts`

Current source inspection still found the Round 21 gaps [S02]:

- Provider streams accumulate text and usage but do not persist full request envelopes, provider request IDs, provider response IDs, or uniform finish/stop status.
- The generic SSE reader ignores the `[DONE]` marker.
- The trace records model route, latency, usage, input/system hashes, and fallback metadata, but not a complete provider-bound attempt receipt.
- Measurement persistence performs separate artifact, attempt, and source upserts. Errors are logged, and the function is designed never to throw.
- A terminal run state is written before measurement persistence is invoked in the synchronous and queued completion paths.

The five Round 21 current-state negative claims were therefore revalidated as current after direct inspection. The two D21-A result claims remained `historical_frozen`, and the two Round 21 benefit/cost claims remained `untested_hypothesis`.

5.4 The key logical distinction

The correct transition was not:

“Round 21 was wrong because the code changed.”

It was:

“Round 21's frozen observation remains valid. Its current-state claims became stale when relevant paths changed, and direct inspection produced fresh evidence that restored those claims at the new commit.”

That distinction is the core of Round 22.

6. Literature map and competing positions

6.1 Provenance-first position

W3C PROV models entities, activities, agents, derivations, revisions, invalidations, collections, and bundles [S03]. SLSA provenance records where, when, and how a software artifact was produced [S04]. These standards support version binding, evidence lineage, and provenance-of-provenance.

Established contribution: a claim can be attached to a named system version and evidence-generation activity rather than floating free of time.

Limit: lineage does not decide semantic validity. A perfectly signed receipt can prove that a test ran against commit X without proving that the test supports claim C or that C remains true at commit Y.

6.2 Assurance-case position

SACM defines assurance cases as auditable claims, arguments, context, and evidence, with explicit relationships and states such as needs-support and defeated [S05]. This is closer to the problem because the object is the claim-evidence argument, not only the artifact.

Established contribution: claims should have explicit context, assumptions, reasoning, evidence, and counter-evidence.

Limit: SACM is a metamodel. It does not supply Enceavra's validity predicates, dependency types, unknown-path policy, or revalidation selector.

6.3 Retest-all position

The strongest simple response to change is to rerun every verification procedure for every current claim. This maximizes recall if the procedures themselves are adequate.

Advantage: minimal dependency-model risk.

Costs: high compute and latency; unnecessary provider calls; review overload; greater exposure of sensitive inputs; and a higher chance that teams bypass the process because it is too expensive.

Retest-all remains the correct fallback when dependency coverage is unknown or the consequence of a miss is unacceptable.

6.4 Selective-regression position

Rothermel and Harrold show that regression-test selection can choose a conservative subset and reduce cost, but safety depends on explicit conditions; precision and universal safety cannot both be perfect [S07]. Follow-up empirical work reports that cost effectiveness varies with program and test-suite design [S08].

Transferable idea: revalidation should select the transitive impacted set, not merely the files that changed.

Non-transferable part: CTAG does not inherit the original control-flow algorithm or proof. Agent-system assurance dependencies include external services, schemas, configurations, and semantic interpretation that ordinary code coverage does not capture.

6.5 Continuous-assurance position

The Evidential Tool Bus work represents assurance workflows as dependencies and incrementally reruns directly and indirectly affected goals [S09]. ENTRUST combines runtime/design-time verification with dynamic assurance cases [S10]. Denney and Pai connect operational safety measurements to assurance artifacts and propose consistency checks [S11].

Established contribution: an assurance case can be treated as a maintained computational object rather than a static document.

Important limitation: the ETB authors explicitly note that workflow specification and validation are manual and that their case study did not prove every individual claim [S09]. That limitation directly applies to CTAG.

6.6 ML-systems position

Sculley et al. identify entanglement, unstable dependencies, undeclared consumers, configuration debt, and changes in the external world as sources of hidden ML-system debt [S12]. Breck et al. argue for broad production tests and monitoring beyond offline model evaluation [S13].

Implication: the dependency universe must extend beyond model code. Provider contracts, routing, data transforms, deployment policy, and consumers can invalidate assumptions.

6.7 Brain-inspired analogy

Sevenster, Beckers, and Kindt report that prediction error, measured through expectancy change in a human fear-learning paradigm, helps distinguish simple retrieval, reconsolidation, and formation of new learning [S14]. This is a narrow empirical result with boundary conditions.

Round 22 extracts one testable engineering analogy:

Do not rewrite every stored claim whenever it is retrieved. Trigger a bounded revalidation process when the current dependency state mismatches the state predicted by the claim's validity record.

This analogy motivates an ablation, not a biological claim. CTAG does not copy neural memory, and software dependency tags are not neurons, engrams, or synapses.

6.8 Synthesis

Position	What it gets right	What it misses for Encevera
Provenance only	Versioned lineage and evidence identity	Whether evidence still supports a claim
Static assurance case	Explicit claims, context, evidence, counter-evidence	Automatic change-triggered lifecycle
Retest all	Conservative coverage	Sustainable cost and selectivity
File watchlist	Cheap and simple	New/transitive paths and external contracts
Filename heuristic	Wider surface than a fixed list	Semantic precision and negative-claim scope
Typed dependency graph	Claim-level selective invalidation	Requires correct, maintained dependency declarations

The literature supports building CTAG as a hypothesis. It does not establish that CTAG will outperform simpler policies in Encevera.

7. Claim and evidence table

ID	Claim	Status	Evidence	Boundary
C22-01	The archive ends at Round 21, so this is Round 22.	Established for inspected commit	Overview, index, registry, publications, manuscripts [S01]	Could change after the inspected commit
C22-02	The post-Round-21 diff contains 127 paths: 83 new and 44 edited.	Established for frozen diff	Lovable <code>get_diff; observed_change_set.json</code> [S02]	Says nothing about semantic impact by itself
C22-03	Seventeen changed paths can affect Round 21 assurance scope.	Manual audit result	Direct inspection and typed manifest [S02]	Not automatically discovered; reviewer judgment may be incomplete
C22-04	A watchlist limited to the three old measurement files detects 0 of those 17 changed paths.	Deterministic result	D22-A	Valid only for this diff and gold set
C22-05	The current provider trace still lacks a complete exact request/response receipt and uniform terminal status.	Revalidated current-state observation	Current provider and measurement source [S02]	Source inspection, not live traffic validation
C22-06	Measurement completeness is still not atomic with terminal-state publication.	Revalidated current-state observation	Current finalization, queue, and measurement source [S02]	Database triggers or external behavior not exercised live
C22-07	Provenance is necessary but insufficient for claim validity.	Established conceptual distinction	PROV, SLSA, SACM [S03-S05]	“Necessary” is for CTAG’s design goal, not every imaginable assurance system
C22-08	Selective revalidation can reduce work under explicit assumptions.	Established externally, conditional transfer	Regression-selection and continuous-assurance studies [S07-S11]	No Encevera production effect established
H22-01	CTAG will reduce stale-claim escape under fixed verification budget.	Untested Encevera hypothesis	Proposed experiment E22-B	Requires prospective evaluation
H22-02	Typed dependencies will outperform filename heuristics on unseen changes.	Untested Encevera hypothesis	D22-A is only authored-fixture support	Manual labels risk circularity
H22-03	Separating historical and current states will reduce misleading archive carry-forward without harming reproducibility.	Untested Encevera hypothesis	Logical design argument	Needs user-facing and archival evaluation
H22-04	Mismatch-triggered revalidation will be more compute-efficient than periodic expiry at comparable stale-claim escape.	Untested brain-inspired engineering hypothesis	Narrow analogy to prediction-error-triggered updating [S14]	Biological transfer is not established

8. Proposed mechanism: Change-Triggered Assurance Graph (CTAG)

8.1 Claim card

Every assurance-relevant statement becomes a structured claim card:

```
claim_id
statement
kind: historical_frozen | current_positive | current_negative | hypothesis
scope
evidence_commitment
system_commit
dependency_tags[]
external_contract_versions_or_inspection_dates[]
validity_predicate
revalidation_procedure
status
```

The public paper may render prose, but the underlying card is machine-readable.

8.2 Claim states

CTAG uses these states:

- `historical_frozen`: valid only as a report about the named frozen evidence and system.
- `current`: valid for the designated current commit without an intervening relevant change.
- `stale_needs_revalidation`: a dependency changed or became unknown.
- `revalidated_current`: fresh evidence satisfied the validity predicate at the candidate commit.
- `contradicted`: new evidence supports the negation or invalidates the claim.
- `superseded`: a newer claim replaces the old proposition without rewriting history.
- `untested_hypothesis`: proposed causal or performance expectation.
- `unknown_scope`: dependency coverage cannot be established.

8.3 Dependency types

The Round 22 prototype uses:

- `provider_boundary`
- `provider_stream`
- `model_router`
- `input_serialization`
- `measurement_collector`
- `measurement_storage`
- `release_path`
- `schema`
- `execution_path`
- `orchestration`
- `external_contract`
- `evidence_semantics`
- `cost_accounting`
- `ingress`

These types are not claimed to be complete. Unknown production-relevant types map to `unknown_execution` and fail closed within the enclosing assurance scope.

8.4 Change event

A change event records:

- old and new system commits;
- changed artifact commitments;
- semantic dependency tags;
- external contract or policy revisions;
- classifier version and human overrides;

- the claim set evaluated;
- selector output;
- revalidation results and evidence commitments.

No customer documents, raw prompts, credentials, or proprietary routing heuristics need to enter the public graph. Content hashes and redacted semantic tags are normally sufficient.

8.5 Expiry algorithm

- 1 Capture the exact old and candidate commits.
- 2 Classify changed artifacts and external contracts into dependency tags.
- 3 Compute the transitive closure from changed tags to current claims.
- 4 Preserve `historical_frozen` claims without modification.
- 5 Move intersecting current claims to `stale_needs_revalidation` before release copy can call them current.
- 6 For any unknown production path, stale all current claims in its enclosing execution scope.
- 7 Run each stale claim's named revalidation procedure under the declared budget.
- 8 Attach fresh evidence and move the claim to `revalidated_current`, `contradicted`, `superseded`, or leave it stale.
- 9 Record selection misses and human overrides as data for later refinement.

8.6 Why negative claims need wider dependencies

A positive claim such as “path P records request ID” can be supported by direct evidence on P. A negative claim such as “the system does not record request IDs” requires coverage of every relevant path. Adding a new provider, custom engine, fallback, queue, or cancellation route can invalidate the coverage even if the old measurement file is unchanged.

CTAG therefore requires negative claims to name the execution-path closure used to establish absence. A new unclassified path immediately makes the negative claim stale.

8.7 External contracts

Providers can change event schemas, finish reasons, IDs, usage fields, retention policy, or endpoint behavior without a repository diff. A current claim may therefore depend on:

- a pinned contract version;
- an inspection date and maximum age;
- a provider changelog commitment;
- a conformance fixture;
- or an observed production envelope schema.

Time-to-live alone is a coarse fallback. CTAG prefers event-triggered invalidation when a reliable provider change event exists and bounded expiry when it does not.

8.8 Release rule

Public current-state copy may cite only claims in `current` or `revalidated_current` for the candidate commit. Historical pages must show their frozen date and commit. A stale claim can remain visible only if clearly labeled stale and not used to support a current product assertion.

8.9 Integrity and independent review

The change classifier, dependency manifest, and revalidation procedure can share a common blind spot. High-impact claims should therefore require at least one independent check:

- a second reviewer of dependency coverage;
- an automatically derived code/runtime relation compared with the authored graph;
- or a controlled mutation that must trigger the claim.

Agreement is a review signal, not proof.

9. Worked example: a new provider and custom engine path

9.1 Before the change

At the Round 21 commit, D21-A reported:

- the provider trace lacked a complete envelope;
- terminal status was incomplete;
- measurement persistence was best effort and post-terminal;
- no complete attempt-bound gate existed.

These were both historical observations and current-state claims at that commit.

9.2 Change event

The later diff adds a new provider route, provider-specific role and reasoning serialization, custom engine profiles, engine execution, queue behavior, public API selection, evidence semantics, and cancellation [S02]. The old measurement collector files are unchanged.

9.3 Naive outcomes

- **No invalidation:** the archive silently presents Round 21's system-wide absence claims as current despite uninspected new paths.
- **Old-file watchlist:** no relevant file appears changed, so no claim expires.
- **Retest all:** every current claim and unrelated paper is rerun.

9.4 CTAG outcome

The change tags intersect all five Round 21 current claim cards:

Claim	Triggering dependency examples	Initial transition
Exact provider envelope absent	provider boundary, serialization, router, execution path	stale
Terminal marker absent	provider stream, external contract	stale
Measurement not atomic	execution path, release path, schema	stale
Release not receipt-gated	release path, execution path	stale
No production ABMG	provider, measurement, release, schema, orchestration	stale

The two numerical D21-A findings remain `historical_frozen`. The hypotheses remain untested.

9.5 Revalidation

Direct inspection of every current provider stream and trace confirms that request/response IDs and uniform terminal fields are still absent. Direct inspection of finalization and measurement persistence confirms terminal state is written first and measurement rows remain separate best-effort writes. Repository-wide inspection finds no atomic receipt gate.

The five stale claims become `revalidated_current` at the new commit with fresh evidence. If even one new execution path could not be inspected, the applicable claim would remain stale.

9.6 What this example does not establish

It does not show that CTAG discovered the right dependencies automatically. A human constructed the tags after inspecting the diff. It shows that the lifecycle semantics can represent the correct distinction and execute deterministic routing once the graph exists.

10. D22-A: deterministic code-change diagnostic

10.1 Purpose

D22-A tests three limited properties:

- 1 Can the proposed claim-state rules preserve frozen historical findings while expiring current claims?
- 2 Can an explicit typed manifest route a real diff more selectively than broad baselines?
- 3 Does the implementation behave as specified on constructed mutations, including an unknown execution path?

It does not test automatic dependency discovery, answer quality, production reliability, security, cost, or latency.

10.2 Inputs

- Actual 127-path diff from Round 21 to the current inspected commit.
- Manually audited 17-path critical set.
- Nine Round 21 claim cards: two historical, five current negative claims, and two hypotheses.
- Thirty-two constructed mutation fixtures covering provider, stream, router, serialization, measurement, release, schema, orchestration, evidence, cost, presentation, tests, external contract, and unknown-path changes.

10.3 Baselines

B0 - Direct measurement-file watchlist. Select only changes to the three original measurement/finalization files.

B1 - Filename keyword heuristic. Select paths containing provider/model/engine/run/queue/measurement/panel terms. On a selected fixture, revalidate all five current claims.

B2 - Retest all. Select every changed path and every current claim.

CTAG-0.1. Use authored dependency tags and claim intersections. Unknown execution paths select all five current claims.

10.4 Execution

The diagnostic is pure Python. It made:

- **0 model calls**
- **0 provider calls**
- **0 production database calls**

Seven internal consistency checks passed, including diff counts, gold-set inclusion, frozen-history preservation, all-five current-claim expiry, unknown-path fail-closed behavior, exact constructed-fixture routing, and the observed measurement-watchlist miss.

10.5 Actual-diff results

Selector	Selected paths	True critical	False positives	False negatives	Precision	Recall
Measurement-only watchlist	0	0	0	17	1.0000*	0.0000
Filename heuristic	37	12	25	5	0.3243	0.7059
Retest all	127	17	110	0	0.1339	1.0000
CTAG authored manifest	17	17	0	0	1.0000	1.0000

Note: Precision is defined as 1 when nothing is selected; recall exposes the total miss.

The CTAG manifest reduced selected-path scope by 86.61% relative to retest-all. This number must not be read as compute savings or effectiveness because the 17-path set and tags were manually authored from the same inspected change set.

10.6 Constructed mutation-suite results

Selector	Exact cases	Claim-edge precision	Claim-edge recall	Event recall	False-alarm events	Mean claims selected per event
Direct file watchlist	9/32	0.8667	0.1667	0.1200	0	0.469
Filename keyword	7/32	0.5889	0.6795	0.6000	3	2.812
Retest all	3/32	0.4875	1.0000	1.0000	7	5.000
CTAG typed dependencies	32/32	1.0000	1.0000	1.0000	0	2.438

These are specification tests on authored fixtures. Perfect CTAG performance is expected if the implementation correctly applies the same manifest semantics used to label the fixtures. It is not evidence of out-of-sample accuracy.

10.7 State-transition result

- Historical claims preserved: 2
- Current claims initially made stale: 5
- Current claims restored after direct inspection: 5
- Hypotheses left untested: 2

10.8 Interpretation

Three useful conclusions survive the limitations:

- 1 A file watchlist can miss an assurance-relevant expansion even when the audited collector itself is unchanged.

- 2 Historical and current claims require different state transitions.
- 3 A typed graph is executable and can express fail-closed unknown handling.

The diagnostic does not support claims that CTAG is complete, cheaper in production, or better than expert review.

11. Competing explanations and alternatives

11.1 “This is just ordinary regression testing”

Partly. CTAG deliberately borrows selective revalidation. But ordinary test selection usually maps changed program elements to tests. CTAG maps heterogeneous changes to claims, evidence, and validity predicates, including provider contracts and negative claims. The transfer must be evaluated, not assumed.

11.2 “A signed provenance graph is enough”

No. It can establish what artifact and procedure produced the evidence. It cannot establish that the procedure remains adequate for the claim or that a new execution path remains within scope.

11.3 “Just timestamp every claim and expire it after 30 days”

Time-based expiry is simple and useful when external change signals are unavailable. It is both under- and over-inclusive: a provider contract can change one day later, while a frozen deterministic claim can remain valid indefinitely. E22-B includes periodic expiry as a baseline.

11.4 “Retest everything after every commit”

This is a defensible high-recall fallback for small systems or high-severity changes. It may also be the only safe option when the graph is incomplete. CTAG must beat it on useful current availability and budget efficiency without materially increasing stale-claim escape.

11.5 “Human reviewers can remember what matters”

Human judgment remains necessary, especially for negative claims and semantic changes. The actual diff demonstrates why memory alone is fragile: provider and execution surfaces expanded across many files while the familiar measurement files remained untouched. CTAG aims to make the review target explicit and auditable, not eliminate reviewers.

11.6 “The five Round 21 claims happened to remain true, so expiry was unnecessary”

That conclusion uses hindsight. Before inspection, the new provider and execution paths could have implemented missing receipt fields or changed finalization ordering. Expiry is about the validity of the evidence link, not whether the claim eventually receives the same truth value.

11.7 “The result is circular”

The strongest D22-A number is circular if interpreted as discovery accuracy: humans curated both the critical set and CTAG tags. The paper therefore treats it only as an executable protocol check. E22-B breaks the circle by freezing the graph before unseen changes and using independent impact labels.

12. Falsifiable hypotheses

H22-1: stale-claim containment

Under equal total verification budget, CTAG will produce a lower stale-claim escape rate than no invalidation, fixed-file watchlists, filename heuristics, and periodic expiry.

Falsifier: CTAG misses as many or more independently labeled affected claims, or its graph is so incomplete that unknown-scope fallback degenerates to retest-all on most commits.

H22-2: selectivity

At matched affected-claim recall, CTAG will revalidate fewer unaffected claims than retest-all and periodic expiry.

Falsifier: false invalidation or revalidation volume is not materially lower, including graph-maintenance labor.

H22-3: negative-claim protection

Execution-path closure and unknown-path fail-closed rules will reduce misses for negative claims compared with direct-file watchlists.

Falsifier: unseen execution paths bypass CTAG or negative-claim coverage cannot be maintained with acceptable reviewer effort.

H22-4: historical integrity

Separating `historical_frozen` from current states will reduce misleading carry-forward while preserving reproducibility and citation stability.

Falsifier: readers become more confused, historical links break, or archives silently rewrite old states.

H22-5: mismatch-triggered efficiency

Change-triggered revalidation will achieve equal or lower stale-claim escape than fixed time-to-live refresh at lower total revalidation cost.

Falsifier: external changes are poorly observable, so time-based expiry catches materially more affected claims at comparable cost.

H22-6: independent derivation value

Comparing authored dependencies with runtime or static dependency evidence will catch manifest omissions that either method alone misses.

Falsifier: the second view adds cost without identifying unique true omissions across the prospective sample.

13. Proposed experiment E22-B

13.1 Objective

Evaluate prospective claim-expiry policies on unseen Encevera changes while controlling total verification compute and preventing the selector authors from labeling their own success.

13.2 Preregistration

Before observing evaluation changes, freeze:

- the claim-card set;
- dependency ontology and CTAG manifest;
- selectors and tie-breaking rules;
- severity weights;
- verification procedures;
- compute accounting;
- primary and secondary metrics;
- missing-data and unknown-path rules;
- stop conditions.

13.3 Evaluation stream

Use two complementary sources:

- 1 **Prospective real commits:** consecutive production-intended commits after the freeze, including no-impact changes.
- 2 **Controlled mutations:** reviewer-authored changes across provider events, fallback, serialization, orchestration, measurement, release, schema, evidence semantics, external contract, UI-only, and unknown-path families.

Do not reveal mutation labels to selector maintainers until predictions are committed.

13.4 Independent gold set

For each change, two reviewers independently inspect the old and candidate systems and label:

- claims that could be invalidated;
- evidence required for revalidation;
- consequence severity;
- whether the change introduces a previously unknown path.

Disagreements go to adjudication. Record inter-reviewer agreement and adjudication rate. The gold set is a review product, not ground truth in a metaphysical sense.

13.5 Arms

- **A0: No automatic invalidation.** Current claims persist until manually noticed.
- **A1: Fixed time-to-live.** All current claims expire at a preregistered interval.
- **A2: Retest all.** Every current claim is selected after every production-relevant change.
- **A3: Direct file watchlist.** Only originally inspected implementation files trigger revalidation.
- **A4: Filename heuristic.** Broad path/keyword rules.
- **A5: CTAG authored dependencies.** Typed graph plus unknown-path fail-closed.
- **A6: CTAG plus independent derivation.** Authored graph compared with static/runtime dependency evidence; union on disagreement.

13.6 Controlled total compute

Each arm receives the same per-change verification allowance:

- identical maximum provider-call count;
- identical maximum input-token and output-token envelope;
- identical wall-clock ceiling;
- identical model/provider versions when model calls are required;
- identical test and tool versions;
- identical source snapshot and cache policy.

Deterministic code checks count by measured wall time and normalized runner resources. Model checks count by actual calls and tokens. If an arm selects more work than the allowance, it must apply a preregistered priority rule and mark the remainder `deferred_due_to_budget`; silent overrun is not allowed. This turns “retest all” into a real fixed-budget baseline rather than an unlimited-compute straw man.

13.7 Primary metrics

- 1 **Severity-weighted stale-claim escape rate:** affected claims still represented as current after the arm's process.
- 2 **Affected-claim recall:** fraction of independently labeled affected claims selected.
- 3 **Useful-current availability:** fraction of unaffected or successfully revalidated claims available within the release window.
- 4 **Budget-normalized verified claims:** successfully revalidated affected claims per provider call, token, and wall-clock unit.

13.8 Secondary and regression metrics

- Precision and false invalidation rate.
- Time from change to stale state.
- Time from stale state to revalidated or contradicted state.
- Unknown-scope frequency.
- Deferred-due-to-budget rate.
- Graph override and adjudication rates.
- Revalidation procedure failure rate.
- Historical-claim mutation incidents.
- Public-copy violations where a stale claim appears current.
- Privacy incidents or sensitive-data expansion.
- Provider cost and p50/p95 release latency.
- Existing answer-quality and safety metrics, to detect a revalidation mechanism that harms the product indirectly.

13.9 Ablations

- Remove external-contract dependencies.
- Remove transitive closure and use only direct path edges.
- Treat negative claims like positive claims.

- Disable unknown-path fail-closed.
- Collapse historical and current states.
- Replace typed tags with filenames only.
- Remove independent graph derivation.
- Replace event-triggered expiry with time-to-live only.

Each ablation must use the same total compute allowance.

13.10 Statistical analysis

Use paired comparisons by change event. Report raw counts, effect sizes, and uncertainty intervals rather than only significance tests. Stratify by change family and claim polarity. Do not pool low-severity documentation changes with provider or release changes without showing both strata. Predefine how multiple claims from one commit are clustered.

No effect size or sample outcome is claimed in this paper.

13.11 Stop conditions

Pause the shadow experiment if:

- a high-severity stale claim escapes into public current-state copy;
- historical evidence is mutated or detached from its commit;
- unknown paths are treated as safe by default;
- sensitive source content enters the public graph;
- or compute accounting cannot be reconciled.

Shadow mode must not block production until the false-invalidation burden and override process are characterized.

14. Expected failure modes

14.1 Missing dependency edge

The dominant risk is false confidence from an incomplete graph. Mitigations include unknown-path fail-closed, second-view derivation, mutation tests, and periodic full audits.

14.2 Over-broad dependency tag

A coarse tag can stale most claims on every change, recreating retest-all. Measure fan-out and require justification for high-degree edges.

14.3 Semantic change hidden in an unchanged path

Environment variables, database functions, provider behavior, model versions, or data contracts can change without a tracked source diff. External dependencies need version or inspection records and bounded expiry.

14.4 Generated-code noise

Schema snapshots and generated routes can create large diffs. CTAG must classify generated artifacts without assuming they are irrelevant; a generated change can still expose a real semantic change.

14.5 Claim wording drift

A claim may be rewritten so narrowly that old evidence appears valid, or so broadly that the graph no longer covers it. Claim changes are themselves revision events and require review.

14.6 Revalidation procedure drift

If a test changes with the system, a passing result may not be comparable. Procedure versions and their own dependencies must be recorded.

14.7 Selector-author leakage

Authors may tune tags to known changes, as in D22-A. Prospective freezing and blinded mutations are required.

14.8 Current-state availability collapse

Fail-closed rules may leave too many claims stale, delaying releases or making the public site uninformative. Useful-current availability is therefore a primary metric, not an afterthought.

14.9 False precision

A detailed graph can look authoritative while containing subjective edges. CTAG should expose reviewer, version, and uncertainty, and should never convert graph completeness into a probability of truth without calibration.

14.10 Privacy and intellectual-property leakage

Dependency graphs can reveal architecture or customer-specific paths. Public artifacts should use redacted tags and commitments; private evidence may retain exact paths under access control.

15. Limitations

- 1 The actual-diff critical set was manually curated by the same research process that authored the CTAG manifest.
 - 2 D22-A tested routing semantics, not automatic dependency discovery.
 - 3 The 32 mutations were constructed, not sampled from independent production incidents.
 - 4 Code was inspected, but no live provider stream, production database, or deployed release flow was exercised.
 - 5 External provider contracts were not exhaustively versioned or conformance-tested.
 - 6 The current-state negative claims may still miss execution paths not visible in the inspected repository or connector view.
 - 7 No model-quality, answer-reliability, security, latency, or cost benefit was measured.
 - 8 The assurance literature is largely safety- and software-engineering oriented; transfer to collaborative AI systems remains empirical.
 - 9 The brain-inspired analogy comes from a narrow fear-learning paradigm and does not establish a general neural algorithm for software assurance.
 - 10 CTAG maintenance may cost more than it saves. E22-B must count authoring, review, overrides, and false invalidations, not only automated execution.
-

16. Implementation sketch without production claim

A minimal shadow implementation should add:

- 1 A versioned claim-card registry.
- 2 A change-event collector bound to old and new commits.
- 3 A typed manifest with reviewer identity and graph version.
- 4 A deterministic selector with unknown-path fail-closed.
- 5 Claim-state transitions stored separately from immutable historical evidence.
- 6 A revalidation queue with fixed-budget accounting.
- 7 A copy guard that prevents stale claims from being rendered as current in preview.
- 8 A dashboard showing stale, deferred, contradicted, revalidated, and unknown-scope states.
- 9 Mutation tests for every negative current-state claim.
- 10 A private/public split that reveals methods and caveats without exposing protected prompts, customer material, or proprietary routing details.

The implementation should begin in shadow mode. It should record what it would have blocked, not block releases, until prospective miss and false-invalidation rates are measured.

17. Round 23 handoff

Round 23 should not run another broad conceptual review. It should implement and evaluate the smallest real CTAG shadow slice while preserving the still-open Round 21 ABMG requirement.

Minimum Round 23 deliverable:

- 1 Encode the five Round 21 current claims and two historical findings as real versioned claim cards.
- 2 Emit change events for provider, execution, measurement, schema, and release paths.
- 3 Mark claims stale in preview before any revalidation.
- 4 Run exact named revalidation procedures and attach fresh commitments.
- 5 Add controlled mutations for a new provider path, ignored terminal marker, changed finalization order, measurement-write failure, and unknown adapter.
- 6 Measure false invalidation, stale-claim escape, useful-current availability, wall time, and review labor.
- 7 Keep the ABMG effectiveness hypotheses untested until the Round 21 attempt-bound receipt is actually implemented.

Stop rule: if the shadow graph cannot catch unknown execution paths or preserve immutable historical claims, do not integrate it with public release labels.

Next research question: Does a prospective CTAG shadow gate reduce stale-claim escape under fixed verification budget on unseen real commits, and which independent dependency view adds unique catches?

18. Reproducibility and artifact inventory

The Round 22 package contains:

- [Enceвра_Research_Round_22.md](#) - this manuscript.

- `Enceвра_Round_22.pdf` - rendered publication.
- `observed_change_set.json` - exact 127-path diff inventory and commit pair.
- `claim_dependency_manifest.json` - Round 21 claim cards, typed changed artifacts, and revalidation states.
- `proposed_protocol.json` - CTAG-0.1 state and release rules.
- `mutation_fixtures.json` - 32 constructed change cases.
- `diagnostics.py` - deterministic diagnostic implementation.
- `diagnostic_results.json` - exact results and limitations.
- `source_ledger.json` - sources, inspection scope, support, and limits.
- `README.md` - reproduction instructions and checksums.

To reproduce D22-A:

```
python output/round22/diagnostics.py
```

The script requires only Python's standard library. It makes no network, model, provider, or database call.

19. Sources

[S01] Enceвра Research archive and manuscripts. Lovable project

8483bc59-da36-492e-b8b8-6bf684070241, inspected at commit 9441b602f7b65528734bd46c5af3a108b3a01a0d, 13 October 2026. Inspected: overview, index, publications, registry, and full Rounds 21, 20, 10, and 9. [Project](#)

[S02] Enceвра source diff and current execution paths. Diff from

311898162201788e8afa4bf5202a36e80f86ffd0 to 9441b602f7b65528734bd46c5af3a108b3a01a0d, inspected 13 October 2026. Full path inventory and inspected-file findings are preserved in the Round 22 package. [Project](#)

[S03] Moreau, L., Missier, P., et al. "PROV-DM: The PROV Data Model." W3C Recommendation, 30 April 2013. Inspected: abstract, introduction, entity/activity/agent model, derivation, revision, invalidation, bundles, and provenance-of-provenance. [W3C](#)

[S04] Supply-chain Levels for Software Artifacts. "Provenance," SLSA specification version 1.2, approved, accessed 13 October 2026. Inspected: provenance definition and build/source distinctions. [SLSA](#)

[S05] Object Management Group. "Structured Assurance Case Metamodel (SACM), Version 2.3." Formal/23-05-08, October 2023. Inspected: scope, structured arguments, evidence, controlled vocabulary, claims, asserted evidence/context, and counter-evidence. [OMG PDF](#)

[S06] National Institute of Standards and Technology. "Secure Software Development Framework (SSDF) Version 1.1." NIST SP 800-218, final 3 February 2022. Inspected: PO.3.3 and PO.4.1-PO.4.2. [NIST](#)

[S07] Rothermel, G., and Harrold, M. J. "A Safe, Efficient Regression Test Selection Technique." ACM Transactions on Software Engineering and Methodology 6(2), April 1997, pp. 173-210. DOI: 10.1145/248233.248262. Inspected: full paper. [PDF](#)

[S08] Rothermel, G., and Harrold, M. J. "Empirical Studies of a Safe Regression Test Selection Technique." IEEE Transactions on Software Engineering 24(6), June 1998. Inspected: author-hosted metadata and abstract. [University of Nebraska-Lincoln](#)

[S09] Sorokin, L., Bouchekir, R., Beyene, T. A., Liao, B. H.-C., and Molin, A. "Towards Continuous Assurance Case Creation for ADS with the Evidential Tool Bus." 4 March 2024. Inspected: full HTML, especially Sections 3.5-4. [arXiv](#)

[S10] Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M. U., Habli, I., and Kelly, T. “Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases.” Submitted 18 March 2017; revised 22 November 2018. Inspected: abstract, methodology, and evaluation scope. [arXiv](#)

[S11] Denney, E., and Pai, G. “Reconciling Safety Measurement and Dynamic Assurance.” SAFECOMP 2024, submitted 30 May 2024. DOI: 10.1007/978-3-031-68606-1_4. Inspected: abstract and framework scope. [arXiv](#)

[S12] Sculley, D., Holt, G., Golovin, D., et al. “Hidden Technical Debt in Machine Learning Systems.” NeurIPS 2015. Inspected: full paper, especially system boundaries, dependencies, configuration, monitoring, and reproducibility. [NeurIPS PDF](#)

[S13] Breck, E., Cai, S., Nielsen, E., Salib, M., and Sculley, D. “The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction.” IEEE Big Data, 2017. Inspected: official publication record and abstract. [Google Research](#)

[S14] Sevenster, D., Beckers, T., and Kindt, M. “Prediction Error Demarcates the Transition from Retrieval, to Reconsolidation, to New Learning.” Learning & Memory 21(11), 2014, pp. 580-584. DOI: 10.1101/lm.035493.114. Inspected: PubMed record, abstract, experimental conditions, and figure descriptions. [PubMed](#)

20. Final claim boundary

Round 22 does not show that CTAG improves Encevera, and it does not show that Encevera has implemented Round 21's attempt-bound measurement gate. It establishes a narrower result: the research program needs an explicit lifecycle for evidence-backed claims, because historical truth and current assurance diverge when the system changes. CTAG is a concrete, testable mechanism for that lifecycle. D22-A shows the mechanism can be executed against the actual diff and fail closed on authored fixtures. Only a prospective, independently labeled, fixed-budget shadow experiment can determine whether it is reliable and worth keeping.