HEARSAY: CAN AN AUDITOR TRUST THE RECORD A DEPLOYED AGENT HARNESS WRITES?

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent harness, the code that turns a model into an agent, writes its own record of each run, and that record is all a later reader gets when a run is disputed, investigated or audited. We call a record *evidentiary* when a reader who was not there can check it without trusting the writer. Across sixteen deployed frameworks, none writes one in full. Hearsay examines the record, not the task: five harnesses run fourteen tasks, three blinded LLM examiners and a human panel read the records, and every excerpt an examiner quotes is checked mechanically for who wrote it. Two findings follow. First, the record lets a reader name the fault but not prove how the run went. Examiners name the right fault in 74 to 91% of 140 runs by a majority of three model graders, and the fault can be proved, but only from two files the benchmark adds, the failing test's output and the final diff; for what happened in between, fewer than one citation in ten lands on anything the harness did not write, and when we delete, rewrite or fabricate entries in copies of the records, the examiner with the fewest false alarms catches half of them. Second, the remedy is a second author, not a stronger seal on the first. An append-only log of what passes between harness and model, kept outside the harness, is read against the record in both directions: a query finds what the log saw and the record lacks, a reverse check what the record holds and the log never saw. Together they report all 28 omissions and fabrications we made a harness commit as it ran, where a hash chain over the harness's own record passes all 28, since it hashes whatever the harness wrote. On 115 records we falsified by editing copies they report 98, each scored against a clean copy of its run with rules written on those pairs; held out on another vendor's records, the reverse check catches 29 of 34 fabrications. Handed the log, examiners keep their fault verdicts but rest more of their citations on what the harness did not write. What makes a record evidence is who writes it, not what is captured.

1 INTRODUCTION

LLM agent harnesses run unattended in production. They record much of what they do (Liu et al., 2026a; Lin et al., 2026), and regulation increasingly requires them to (Chan et al., 2024; European Parliament and Council, 2024). A growing literature reads those records to explain failures: benchmarks locate the failing step (Zhang et al., 2025; Deshpande et al., 2025), taxonomies classify it (Cemri et al., 2025), and fuller traces improve attribution (Chen et al., 2026). However, every such record is written by the running system, for the people present while it ran. The reader who later disputes, investigates or audits the run was not there, and cannot check it without trusting the system that wrote it. Whether a record can serve that reader is a property of the record, not of the task.

We call a record *introspective* when it holds only the system's own account of its execution, and *evidentiary* when a reader who was not there can check the run from it without taking the system's word, including that something did *not* happen. We name the benchmark after the hearsay rule: a party's own out-of-court statement, offered in its favour, cannot establish its truth. Operationally, a claim counts only where a witness the system does not control confirms it.

The records deployed harnesses write are not evidentiary. Of sixteen frameworks we audit, none writes a record an absent reader could check, while, by our instrument, the properties that serve the people present at a run are fuller the more active a project is (§3). Two gaps have kept this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

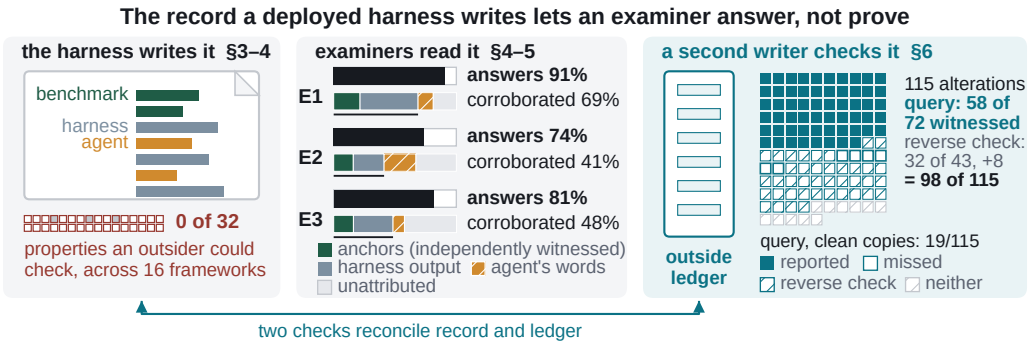


Figure 1: One record, three readers, one outside witness. Left: across sixteen frameworks no property an outsider could check is fully present, three cells partial (§3). Centre: under the native record each examiner names the fault often (majority of three graders, §5.1), yet little of what it cites was written outside the harness, the green segment (§5.2; colours as in Fig. 3). Right: of 115 edited copies, taken as records false from the start, the query reports 58 of the 72 it witnessed, the reverse check 32 of the other 43 and 8 of the query's misses, 98 in all, each paired with a clean copy of its run (§6).

unmeasured (§2): (1) **no benchmark has the record as its object**, and (2) **evaluations grade the judge's verdict, not what it rests on**: nothing checks who wrote the text a judge cites (Zhuge et al., 2025; Gurram et al., 2026).

To close both gaps, Hearsay puts the record, not the task, under examination (Fig. 1). Five harnesses run fourteen tasks twice, 140 runs, in a sandbox that records model calls, processes and files independently of the harness. Three blinded LLM examiners from three vendors (E1–E3) read each record under progressively more material, and three grader models score them. For every excerpt an examiner quotes, we check mechanically who wrote it: the agent's own words (*narration*), output the harness recorded or material from outside it (*corroborated*: not the agent's say-so, though the harness may have filed it), and, of that, material the harness did not write (*independently witnessed*). Under the native record the only such material is the benchmark's two *anchors*, the failing test's output and the final diff; §6 adds an outside log.

Two results follow. **The native record lets an examiner answer, not prove.** Examiners locate the fault in up to 91% of runs, yet the harness itself wrote most of what they cite, and when we alter copies of the records they catch at most half of the alterations at a usable false-positive rate (§5). **The remedy is another writer, not richer logging.** A small append-only log outside the harness does two jobs. Its copy of the record, kept as it is written, exposes any later edit, as a seal would. Its record of what passed between harness and model catches the writer, which no hash chain over the harness's own record can: read against the record in both directions, it reports what the harness left out and what it made up. Handed the log, the examiners keep their verdicts but rest more of their citations on what the harness did not write (§6).

We make four contributions. (1) It **defines evidentiary status** for a record and **builds an instrument** that grades each citation by who wrote the text it lands on (§4.3). (2) It **audits sixteen deployed frameworks** (§3). (3) It **shows that examiners answer without proving**, across examiners, and for E1 under every grader, though on another vendor's runs accuracy falls and the witnessed share rises (§5, §7). (4) It **adds one outside writer** to the same runs (§6). We release the benchmark, records, alteration pairs and preregistration.

2 Evidentiary Records, and the Gap in the Literature

We treat a harness's account of its run as a claim, not evidence, like a model's account of its reasoning (Turpin et al., 2023). A record falls short when the harness knew something and did not file it, could not see it, or filed a falsehood. §6 measures the first and third; the record alone cannot show which.

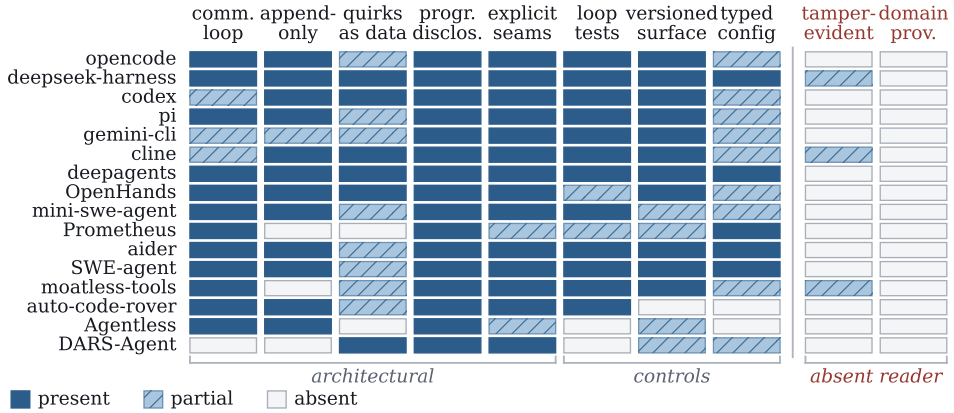


Figure 2: By the instrument, the properties that serve the people present fill in as a project gets busier; the two that serve an absent reader stay empty. Rows: the sixteen frameworks, ordered by commits in the year to 2026-08-25. Left of the rule, the eight present-serving properties; right of it, the two an outsider could check: 0 present, 3 partial, 29 absent. Full column names: Appendix A.

Who&When (Zhang et al., 2025) takes the log as given, while its successors plant faults under replay or generate scenarios (Zhang et al., 2026; Wang, 2026), and taxonomies read the trace (Cemri et al., 2025; Deshpande et al., 2025). Solozobov (2026) poses the question on synthetic, normalised records with a per-record label, whereas our label is per citation on the native record. Outcome benchmarks grade state (Yao et al., 2024; Gurram et al., 2026); LLM judges are scored on verdicts and carry self-preference (Zheng et al., 2023; Panickssery et al., 2024).

On the record itself, the harness literature has no axis for who writes the record (Rombaut, 2026); a study of a flagship commercial harness finds its records "not yet externally auditable" (Liu et al., 2026a), and a survey of 70 harness projects finds no audit trail written outside the runtime (Wei, 2026; Nian et al., 2026). Tamper-evident logging keeps a logger honest about what it stored, not about what happened (Schneier & Kelsey, 1999; Crosby & Wallach, 2009). The design that does capture what happened, one record per party and a check that pins each deviation on a party, is accountability in distributed systems (Haeberlen et al., 2007; 2010); Zheng et al. (2025) already place a kernel-level (eBPF) observer beside coding agents and are the nearest prior system to §6, which adds the check that sets that observer's record against the harness's (Appendix I). Prior work, then, grades the verdict and takes the record as given; whether a deployed harness writes a record that could be checked at all is the question §3 puts to sixteen frameworks.

3 The Gap in Deployed Harnesses

Whether deployed harnesses write records that clear the bar of §1 has not been measured. We audit sixteen agent frameworks at commits pinned between December 2024 and August 2026 (Table 4), with claude-sonnet-5 as the instrument. Every score must quote file-and-line evidence and is discarded when it cannot be found; each verdict is the majority of three blinded passes (Appendix B).

None of the thirty-two cells of external verifiability is `present` (Fig. 2). *External verifiability* puts §1's definition into practice in two properties: a tamper-evident record, checkable without trusting the runtime, and domain provenance, tool results that carry their as-of time, entitlement and cost as first-class fields. The other eight are five architectural properties (a commoditised loop, an append-only replayable session record, model quirks as data, progressive disclosure of context, explicit seams) and three controls (automated loop-test coverage, a versioned public surface, a typed configuration schema), which ordinary engineering produces; they check the instrument, not the claim (Appendix A). Three of the thirty-two are `partial`, all on the tamper-evident record; twenty-nine are `absent`. On a five-rung integrity ladder scored afterwards, from a missing or rewritable record (rung 0), through one append-only writer, a second writer and an outside timestamp, to several parties, fourteen frameworks sit at rung 0, two at rung 1 and none higher (Table 5).

By the instrument, the present-serving properties track activity and the absent-reader ones are empty regardless. The eight properties that serve a present reader average 80% across the sixteen, and that mean rises with commit count, Spearman's $\rho = 0.55$ (Fig. 6 in Appendix B). The two that serve an absent reader stay empty from the most active project to the least. The controls fill in like the architectural properties, so the instrument cannot tell architecture from activity; it reaches `present` readily where nothing is at stake, so the zero is an absence, not strictness.

Three further checks stand behind the zero, and none moves an absent-reader cell to `present` (Table 3 in Appendix B): a critic on a different model from the same vendor read all 77 `absent` cells, at these commits and up to four earlier ones per framework, and five of the 46 it sent back flipped; the model the preregistration named scored nine of the frameworks; and a second instrument from another vendor scored every cell. A fourth falls short: two human annotators re-judged 33 other cells blind (controls it scored `present`, architectural properties it scored `absent`). Where the two disagree, 7 cells, the registered rule scores the cell `partial`, which none of them is in the grid; so scored, they agree with the instrument on 26, under the registered 80%, and the activity pattern above is the instrument's reading (each annotator alone agrees on 29 and 28; Appendix B). The problem of §1 is therefore not hypothetical: none of the sixteen writes an evidentiary record. The preregistration (Appendix H) predicted moderate-to-high reconstruction and fault attribution, low corroboration and near-chance alteration detection, which §4 and §5 test.

4 Benchmark Design

Hearsay examines the record, not the task: can an examiner who was not there reconstruct the run and name the fault, what does that answer rest on, and can it tell when the record was altered (§4.3)?

4.1 Tasks, Harnesses and Runs

The fourteen tasks are of two kinds. Eight are failure archetypes from production incidents, re-implemented as fresh minimal repositories with synthetic data. Six are SWE-bench Verified instances chosen by a mechanical rule fixed before any run; three of them entered after registration, with keys written after the examinations from the upstream gold patches (Appendix C).

Five harnesses run them, all among the sixteen of §3 and at the same pinned commits: the attended products aider, cline and opencode, and the unattended scaffolds SWE-agent and mini-swe-agent. The attended products run unattended here: aider with every confirmation pre-answered, so its records are single-turn, and cline and opencode with auto-approval (Appendix E). Each harness runs every task twice on one backing model (`claude-sonnet-5`), at most seven minutes a run: 140 runs, the *main grid*. The cross-vendor runs repeat every task on `deepseek-v4-pro` (Appendix G), and a write-time test wraps aider's record writer for 28 further runs (§6).

4.2 Record Provenance and the Second Writer

Every run also leaves a record the harness did not write: a recording sandbox watches two boundaries the harness does not control: the **model boundary**, all model traffic through a logging proxy, and the **environment boundary**, processes and files. Four writers leave material for the examiner (Fig. 3a). The harness writes its *native record*. The benchmark writes the two *anchors*, the failing test's output before the run and the final diff. The sandbox writes the *external capture*, and the second writer the *ledger*. The *decision list* is every process run, model call, retry, file write and deletion the capture recorded, listed by fixed rules; reconstruction is scored against its actions on the task's files, and the list itself is handed over only in the richest condition, *ceiling* (§4.3). A per-task key grades the fault.

A hash chain a harness computes over its own record proves only that the record was not revised later, not that it was true. The record is written by the party whose account is in question, so what is missing is a *second writer* (§6), built on the capture. A harness-side writer copies and hashes each increment of the session record outside the harness's execution path; that copy exposes any later edit to the record by construction, which we checked on all 115 edited copies of §5.3 (Appendix G). An environment-side writer adds what crossed the boundaries: model texts, tool arguments and, added after the main-grid bundles were extracted, tool outputs. Both feed one append-only, hash-chained *ledger*, its head timestamped (RFC 3161). A *unit* is one assistant text, tool argument or tool output

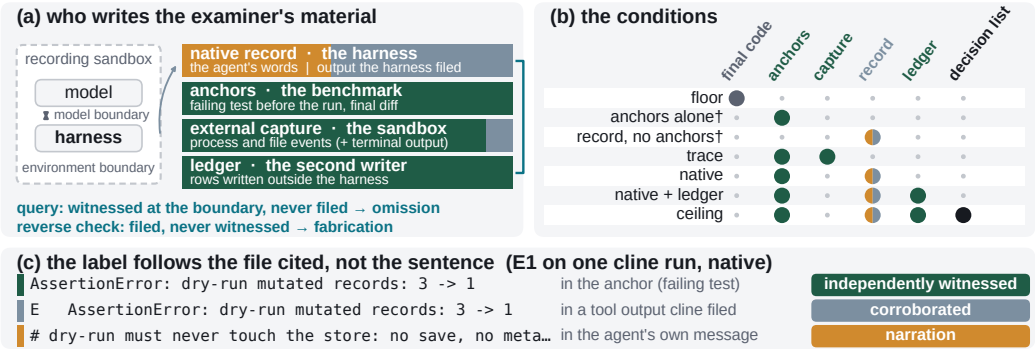


Figure 3: Provenance of the examiner's material. (a) The harness writes its record, partly the agent's own words (orange) and partly output it chose to file (slate); the anchors, the external capture and the ledger are written outside it (green; the capture also carries the harness's terminal output, slate); the second writer's two checks (teal) set ledger and record against each other. (b) What each condition hands the examiner; the half-orange, half-slate dot is the record, which holds both (Appendix D). (c) One sentence E1 cited three times on a cline run: the label follows the file, not the wording.

as sent to the model. A **query** reports witnessed units the record lacks: omissions. A **reverse check** reports record units never witnessed: fabrications. The reverse check first splits the record into units; each harness stores its record differently, so each format needs its own rule, written for all but SWE-agent (Appendix G). The whole is ∼230 lines of standard parts, plus those rules (Haber & Stornetta, 1991; Adams et al., 2001; Dapr project maintainers, 2026; Haeberlen et al., 2007; 2010); the combination and its measurement are new.

Threat model. The adversary is the harness: it may omit, invent or alter what it files, while running or afterwards. It is assumed not to write environment-side rows or bypass the proxy. On the main grid neither is enforced: the writers share one database role and the harness is not confined to the proxy, so a harness breaking them would defeat the witness silently. The write-time test enforces both, with the same result (Appendix G). The operator, who runs both writers, is trusted (§7).

4.3 Examination Protocol and Scoring

Three questions. Each examiner is handed the record of one run and asked three things; we score the record, not the task. **Q1, reconstruction**: the ordered steps, aligned mechanically with the decision list. **Q2, attribution**: what fault the agent addressed and whether its change fixed it, against the key. **Q3, alteration**: whether the record was altered. Clean and altered copies go to separate sessions; detection is read against the examiner's false-positive rate on clean copies (§5.3).

What the examiner is handed is the controlled variable (Casper et al., 2024). Seven *material conditions* run from the final code (*floor*) to record, ledger and decision list (*ceiling*) (Fig. 3b). Floor against native gives what record and anchors add, and the two added conditions (†) separate them. Trace swaps the record for the capture. Native+ledger and ceiling add the ledger, then the list.

Three examiners, one per vendor: E1 (claude-sonnet-5) and E2 (gemini-3.1-pro-preview), frozen before examination, and E3 (deepseek-v4-pro). E1 also drove the runs, a possible self-preference (§7). A *cell* is one run under one condition read by one examiner. Thirty-seven annotators, each on one task and blind to every result and base rate, examine samples under the same protocol (§5.4).

Grading. Three grader models from three vendors, blind to the condition, grade Q2 *correct*, *partial* or *wrong*, and a run counts as correct when two agree. Grader 1 (claude-haiku-4-5, E1's vendor, preregistered) is reported beside the majority; graders 2 and 3 are deepseek-v4-pro and gpt-5.2.

Labels. What a citation is worth is not the examiner's to decide. Its label follows who wrote the text it lands on (Fig. 3c), under rules fixed per record format (Appendix E). *Located*: the excerpt appears verbatim in the file it cites. *Corroborated*: located on output the harness filed, on the anchors or on the ledger. *Independently witnessed*: corroborated on text the harness did not write, the anchors or

	E1	E2	E3		E1	E2	E3
Fault attributed (% of runs)				*Citations under native (% of each examiner's)*			
floor	17	14	3	located verbatim	96.0	92.8	85.0
anchors alone†	64	–	39	corroborated	68.8	41.1	48.4
record, no anchors†	91	–	–	indep. witnessed†	**21.5**	**16.0**	**16.1**
trace	86	70	73	self-reported corroborated	86.7	96.2	94.0
native	**91**	**74**	81	*Corroborated share, other conditions (%)*			
native+ledger	89	72	80	trace	48.6	55.2	45.7
ceiling	92	80	87	native+ledger	74.1	40.6	46.5
native, grader 1	97	76	83	native+ledger, indep. witnessed†	33.9	21.2	21.3
native, other vendor	83	–	69	native, other vendor	65.4	–	43.8
reconstruction recall, median	0.40	0.33	0.40	native+ledger, other vendor	65.9	–	47.5
Alterations (% of copies)				*Ledger shift, native to native+ledger (points, paired)*			
altered copies detected	48.7	85.2	48.7	indep. witnessed†	+12.4	+5.2	+5.2
clean copies flagged	5.2	41.7	20.0	corroborated (registered rule)	+5.3	−0.5	−1.6

Table 1: What the record supports, by examiner (§5); † added after registration. Left: how often each examiner names the fault per condition (§4.3), by the majority of three graders, the preregistered one in its own row. Right: what the examiners' citations rest on, and how far the ledger moves it. A dash marks a condition an examiner did not see; other vendor: 140 deepseek-driven runs; E1 floor: 105 valid cells; E3 native: 118, ceiling: 135; strict aider rule: E1 native 63.1 (App. E); recall is a fraction. Cell counts, intervals (paired over runs; task-clustered in App. F), the narration reading and the other vendor's shifts are in App. F and Table 28; alterations by type in Table 23.

the ledger. *Narration*: the agent's own words. A label grades the file actually cited, so it reflects the examiner's choice as well as the material (§5.2). A ledger row of the agent's own words is a witness to the saying, not to the said: it counts for reconstruction (Q1) but not for the fault (Q2).

After registration. Procedures were fixed before examination; later changes are marked † and dated in Appendix H. Added afterwards, and exploratory: the independently witnessed share, the majority grading rule (adopted after the gradings were seen), the 10% false-positive threshold, two conditions and E3. Two landing rules were later tightened, both lowering a share (Appendix E).

5 Evidentiary Value of the Native Record

The native record lets an examiner diagnose a run but not prove it, a dissociation between accuracy and corroboration. *Accuracy* asks whether the examiner names the fault (Q2); *corroboration* asks what the answer rests on, by the labels of §4.3; *detection* asks whether it notices alteration (Q3).

5.1 Any record lets the examiner name the fault; reconstruction stays sparse

Accuracy (Q2): with the record the examiner names the fault; with the final code alone it does not, and for E1 the anchors add nothing on top of the record (Table 1; Fig. 4a). By the majority of three graders, E1 goes from 17% with the final code alone to 91% with the record, with or without the anchors; the anchors alone give 64%, and for E1 the ceiling adds nothing. The shape holds for E2 and E3, whose ceilings add six points (E3's paired gain is +8.0), under graders 1 and 3; grader 2, the strictest, puts E2 and E3 at 49% and 59% under native (Appendices F, G). Reconstruction (Q1) recall under native has a median of 0.33 to 0.40 at precision 1.0, below the preregistered moderate-to-high (Appendix H): sparse edit and run steps, not invented ones.

5.2 The answer rests on the harness's own account

Corroboration: what the answer rests on is mostly the harness's own account. Under native, 68.8% of E1's citations, 41.1% of E2's and 48.4% of E3's are corroborated, and only 21.5%, 16.0% and 16.1% independently witnessed, pooled over Q1 and Q2 (Table 1; Fig. 4b); that label was added after registration (Appendix H). Every independently witnessed citation lands on the two anchors, the only material the harness did not write, so under native this share is set by the design: it measures how little else an answer could rest on. Even crediting every record citation whose excerpt an anchor also holds verbatim, the shares are at most 30.8%, 21.1% and 21.1% (Appendix F). The rest of the

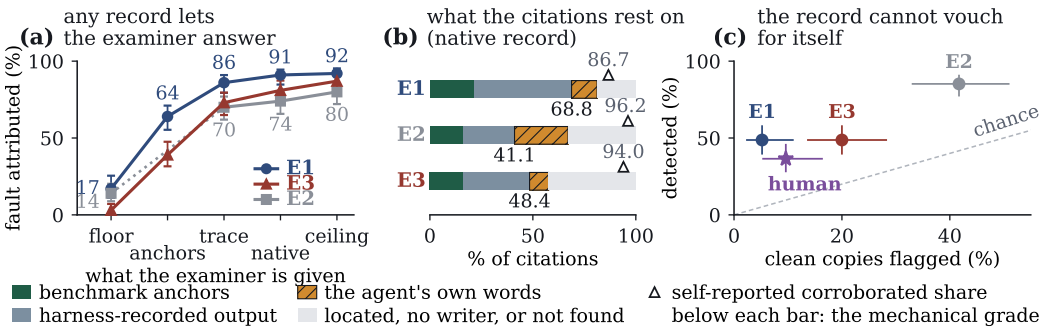


Figure 4: The native record lets an examiner answer, not prove, and cannot vouch for itself. (a) How often each examiner names the fault as the material grows from the final code to the ceiling of §4.3 (majority of three graders over 140 runs, 105 to 140 valid cells, 95% Wilson intervals; App. F); E2 was not run on anchors alone, so its floor-to-trace link is dotted. (b) Where each examiner's citations land under the native record: the green segment is the two anchors, the independently witnessed share (21.5 / 16.0 / 16.1%); with harness-recorded output it makes the corroborated share, below each bar; the agent's own words are narration; the rest is located but attributable to no writer, or not located; above each bar, what the examiner claimed. (c) Over the 115 alteration pairs, how often each reader flags an altered copy against how often it flags a clean one (95% Wilson intervals), the diagonal being chance; the star marks the human panel (§5.4). By type in Table 23.

corroborated citations is tool output the harness recorded or, for aider, status lines its log prints; scored as narration, those lower E1's pooled share by six points and aider's own from 66% to 26% (Appendix E). **Under the external trace with the anchors, E1 attributes the fault nearly as well as under native, 86% against 91%, though its citations verify less often** (61% against 96%; Table 27). Either the capture or the record thus adds over twenty points to the anchors alone (64%).

The fault is provable, but from two lines the benchmark wrote, not from the record. By question, the independently witnessed share is 49 to 70% for the fault and 2 to 9% for reconstruction (Appendix F). For E1, one fault citation in four lacks corroboration, and three quarters of the rest land on the two anchors. Here the two axes come apart: the fault is proved by the failing test's output and the final diff, which any test harness produces, and the account of what happened is not proved.

5.3 The record cannot answer for its own integrity

Alteration detection (Q3): only E1 and, at 9.6%, the human panel keep false positives below 10%, and neither detects more than half of the alterations (Table 1; Fig. 4c). An alteration (a deletion, a rewrite or a fabrication) is injected into each of the 115 records that contain an alterable unit (a message or a tool output). The injector must first pass a gate: shown both copies and asked which was edited, E1's own model must not pick the edited one significantly above chance. It passed on the fifth revision, 65 of 113 parsed answers (57.5%, p = 0.066; earlier rounds 64.5–83.7%; Appendix E). So E1's rates below are for edits its own model did not pick out above chance; a second vendor's judge picked them out at 60.9% (p = 0.012), so for that judge the gate did not hold, though under the 65% ceiling registered for that judge before it ran (Table 30). E1 misses half of the alterations at a 5% false-positive rate. E2 detects 85% only by flagging two in five clean records as altered. Every examiner detects above its own false-positive rate, so the registered prediction of chance-level detection was missed (Appendix H); the 10% threshold is ours, set after the results.

A clean record can be wrong, not merely thin. Two were, with no injection. In one, the record reported normal completion while the seven-minute bound had failed to stop it: the harness could not see that (Appendix G). In the other, a framework's session record contained an assistant message the model did not produce: the harness filed falsely. Neither is visible from inside the record.

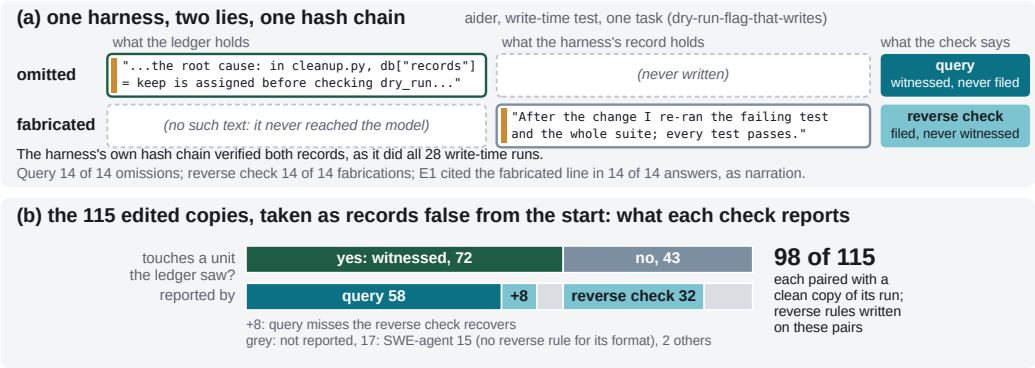


Figure 5: What the second writer reports. (a) Two write-time runs of one task: the ledger holds a reply the harness never wrote (the query reports it), and the record a claimed test run never sent to the model (the reverse check reports it); the harness's own hash chain verified both. (b) The 115 edited copies, taken as records false from the start: which touch a witnessed unit, and which check reports them, each scored against a clean copy of its run (unpaired costs in the text).

5.4 Examiners and people

The examiners disagree sharply under identical inputs, because they cite differently. The gap between E1 and E2 in corroborated share is **+27.7 points [+21.8, +33.3]**, with E3 between them (Fig. 4b). The gap is in reconstruction citations, where E2's land on the agent's own words and E1's on recorded output; on the fault question the two corroborate alike (Appendix F).

Self-report: all three examiners overstate how much of their own evidence is corroborated (Table 1). Each also labels its own citations; no label is used in scoring. E1 calls 86.7% of its native citations corroborated where the rule finds 68.8%; E2's gap is 55 points, E3's 46.

People are not the way out; the bound is the record's. Blind annotators agree with the landing-site rule on 82 of 100 sampled citations; of the 18 others, 7 call a rule-corroborated citation narration, 1 the reverse, and 10 are excerpts they could not find. Twenty annotators, three per copy, read all 115 alteration pairs. By majority they detect 36.5% at 9.6% false positives, and no annotator passes the 60% at 10% registered for a smaller design. Five annotators reading one native run per task attribute the fault in 11 of 14 by majority, 41 of 70 sheets. On that question 56% of their citations land on an anchor, all on the final diff, beside 49 to 70% of the models' fault citations (every human excerpt was found verbatim at its locator). Three annotators grading 100 sampled answers agree with the majority of three graders on 84 but grade more strictly: 12 of 25 native answers correct, against 18, so the model-graded accuracies may run high (Appendix F).

6 Adding a Second Writer

A seal cannot stop the writer. The ledger's copy of each record exposes a later editor (§4.2), but the record is written by the party whose account is in question, and only a second writer checks it.

In the write-time test, a harness that writes falsely is caught by the second writer in both directions, not by its own hash chain (Fig. 5a). In 28 further aider runs (§4.1), half never wrote the model's reply and half claimed a test run that never happened; the harness's own hash chain verified all 28, the two checks reported all 28, and E1 cited each fabrication as narration.

Only the second writer shows what was never filed: on clean runs the query flags only units the harness never filed as sent, from runs that crashed, timed out or misfiled and from tool calls SWE-agent drops.

The second writer is a witness, not a detector (Fig. 5b; Table 2). It reports what crossed the boundary, assistant texts and tool arguments in the ledger the examiners saw, so an edit is *witnessed* when it touches one of those. Each of the 115 edited copies, fed to the harness-side writer as if the

The harness…	Caught by the second writer? Measured
edits the record after it is filed	yes, by the ledger's copy of the record, by construction: 115/115
omits a unit it sent or received	yes, by the query: write-time 14/14; on copies 58/72
files a unit never sent or received	yes, by the reverse check: write-time 14/14; on copies 32 of the 33 with a rule (written on them); held out 29/34
rewrites a tool output it filed	only with a reverse rule: 0/4, all on SWE-agent, which has none
alters one of two copies of a unit it filed	partly: 12 of the query's 14 misses; 7 of the 8 the reverse check recovers
tells model and record a failed tool passed	no: a program's run is witnessed, not its output; not tested
writes as the user, or adapts to the rules	no: out of scope; not tested
evades the proxy or forges witness rows	only under enforced isolation; no evading harness tested
is the operator	no: the operator is trusted

Table 2: What the second writer attests, and what it does not.

harness had filed it, stands in for a record false from the start. Scored against a clean copy of each run, the two checks report 98: the query most edits to witnessed units, the reverse check most of the rest. The reverse rules, written on these pairs, still catch 29 of 34 fabrications held out on another vendor. The cost is the clean twin: read alone, the reverse check flags half the clean copies (52 of 95; the query 19 of 115).

It changes what a verdict can rest on, not the verdict. Fault attribution moves by 2.1 points or less for every examiner (Table 12), while more citations rest on material the harness did not write: +12.4 points for E1 and +5.2 for E2 and E3. **The registered primary outcome, the corroborated share, rises for E1, +5.3 points**, counting ledger rows of the agent's own words as the registered rule does; those rows show what the agent said, not that it was true, and without them it does not rise for E1 (-1.4) and falls on the other vendor's records (Table 29).

7 Discussion and limitations

None of the sixteen frameworks has a second writer. Our explanation: the properties they have serve those present at a run, who file its issues; the reviewer, disputant and regulator are not there.

What each headline rests on. Each has a second check run after registration (Table 30). The held-out reverse check misses its registered bar, and on another vendor's runs our post-hoc dissociation test (accuracy at least 70%, witnessed share at most 25%) fails: E1's witnessed share is 31%, E3's accuracy 69%. Two vendors recur: E1 drove the runs it examines and grader 1 is E1's vendor; four of the five second checks rest on E3's model, also grader 2. On 140 runs driven by another vendor E1 locates the fault less often, 83% against 91%, vendor-dependent by the registered rule, under which a drop whose interval reaches -10 points counts (-7.9 [-15.7, +0.0] points).

The study is five harnesses, fourteen tasks, three examiners and one trust domain; the write-time test is one single-turn harness. What the second writer cannot attest, and what was never tested, is in Table 2; its timestamp fixes only that an increment existed.

8 Conclusion

A record written by the party whose account is in question is testimony. Of sixteen deployed frameworks, none writes one an absent reader could check; handed such a record, examiners name the fault but cannot show how the run went, and catch at most half of its alterations at a usable false-positive rate. A seal catches a later editor; only a second writer outside the harness catches the writer. That no framework has made it is, on our reading, not a matter of capability: the reader it would serve was never in the room.

AI use statement

Large language models enter this work in three roles, which we separate. As **subjects**: the five harnesses under study were run with a single backing model on the main grid, a second on a subset

and a third on the cross-vendor runs, and the records they left are the object of measurement. As **instruments**: the examiners that read those records are models, two named and frozen before the first examination and a third added after registration as a robustness check (§4.3). Every citation score is anchored to a mechanical rule (landing site, paired clean controls) and every fault verdict is read by three graders from three vendors, not by the examiner's own judgement, for the reasons given in §5.4 and §7. As **tools**: LLM-based coding assistants were used to implement the sandbox, extraction, injection, and scoring code, including the seed scripts that generate the synthetic data of the eight prototype tasks; to run the experiments; and to speed up the iteration and revision of the text among the co-authors. A language model was also used to anonymise the human-annotation packages; the annotators themselves worked without AI tools. All code was reviewed by the authors, every reported number was recomputed from the released artefacts, and the authors take full responsibility for the content.

Ethics statement

No employer code or data enters the benchmark: the eight incident-derived tasks are re-implemented as fresh minimal repositories with synthetic data reproducing the failure mechanism only. The alteration experiment (§5.3) manipulates copies of records produced inside our own sandbox, and no third-party record was altered. Examination sent the benchmark materials (synthetic tasks and SWE-bench-derived repositories, no private data) to the APIs of four vendors: three examiners, one per vendor, and a grader from a fourth. The paper makes no claim about the intent of any framework's authors: the absence it measures is explained as an incentive artefact (§7), not as negligence.

Reproducibility statement

Examiners: E1 = `claude-sonnet-5` (also the backing model of the main grid's 140 runs; the cross-vendor runs of Appendix G are driven by E3's model), E2 = `gemini-3.1-pro-preview`, E3 = `deepseek-v4-pro`; graders: `claude-haiku-4-5` (grader 1, fixed in the preregistration), `deepseek-v4-pro` (E3's model), `gpt-5.2`, the headline being their majority; audit instrument (§3): `claude-sonnet-5` with rubric version 2, the wording of Appendix A as handed to every pass, one clause added afterwards and disclosed there, the `claude-fable-5` set of the preregistration being incomplete and used as a cross-instrument check (Appendix B); E2's vendor is distinct from every model under test and every grader; E3's model also drives the cross-vendor runs and serves as the second grader, which §7 and Appendix G account for. Code, data, prompts, grader verdicts, and the preregistration are released with the anonymised repository below; the licence accompanies the release. The design was frozen in a preregistration before any run (its entry of 2026-08-28 froze the benchmark design and its keys in the preregistration's appendices M, N and O; the alteration design was signed 2026-08-29), and deviations are recorded as dated, signed entries. Extraction, folding, injection, and scoring are mechanical scripts. Original artefacts ship alongside every examiner bundle and are tied by SHA-256 to a corpus index. The alteration copies are deterministic functions of the released materials and the injection script, with the record hashes sealed in a manifest. Runs are stochastic. The examination is reproducible from the released bundles; the audit grid of §3 rebuilds from its archived passes (Appendix B). Data, code, and the preregistration are released at an anonymised repository (`https://anonymous.4open.science/r/hearsay-4E7C`), and the de-anonymised archive replaces it on acceptance.

References

Carlisle Adams, Pat Cain, Denis Pinkas, and Robert Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). Technical Report RFC 3161, IETF, 2001.

AgentOps, Inc. AgentOps. `https://agentops.ai`, 2026. Hosted agent observability platform; docs accessed 2026-08-30.

Adam Bates, Dave Tian, Kevin R. B. Butler, and Thomas Moyer. Trustworthy Whole-System Provenance for the Linux Kernel. In *Proceedings of the 24th USENIX Security Symposium*, pp. 319–334, 2015.

Stephen Casper, Carson Ezell, Charlotte Siegmann, et al. Black-Box Access is Insufficient for Rigorous AI Audits. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2024. arXiv:2401.14446.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, et al. Why Do Multi-Agent LLM Systems Fail? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2503.13657.

Alan Chan, Carson Ezell, Max Kaufmann, et al. Visibility into AI Agents. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2024. arXiv:2401.13138.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. Seeing the Whole Elephant: A Benchmark for Failure Attribution in LLM-based Multi-Agent Systems. arXiv:2604.22708, 2026. ACL 2026 (long), pp. 19888–19905.

Scott A. Crosby and Dan S. Wallach. Efficient Data Structures for Tamper-Evident Logging. In *Proceedings of the 18th USENIX Security Symposium*, pp. 317–334, 2009.

Dapr project maintainers. Dapr v1.18 is now available. `https://blog.dapr.io/posts/2026/06/10/dapr-v1.18-is-now-available/`, 2026. Released 2026-06-10; section "Workflow History Tamper Detection" (opt-in, disabled by default).

Darshan Deshpande, Varun Gangal, Hersh Mehta, et al. TRAIL: Trace Reasoning and Agentic Issue Localization. arXiv:2505.08638, 2025.

George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza A. Basrai, and Peter M. Chen. ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

European Parliament and Council. Regulation (EU) 2024/1689 (Artificial Intelligence Act). Official Journal of the European Union, L, 12 July 2024, 2024. Articles 12, 19, 26(6); Recital 71.

Bhaskar Gurram et al. Auditing Automated Evaluation, Error Propagation, and Runtime Mitigation in Tool-Using Language Agents. arXiv:2604.16706, 2026.

Stuart Haber and W. Scott Stornetta. How to Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2):99–111, 1991.

Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. PeerReview: Practical Accountability for Distributed Systems. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP)*, pp. 175–188, 2007.

Andreas Haeberlen, Paarijaat Aditya, Rodrigo Rodrigues, and Peter Druschel. Accountable Virtual Machines. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.

Vishal Karande, Erick Bauman, Zhiqiang Lin, and Latifur Khan. SGX-Log: Securing System Logs with SGX. In *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIA CCS)*, pp. 19–30, 2017.

LangChain, Inc. LangSmith. `https://www.langchain.com/langsmith`, 2026. Hosted LLM-application tracing platform; docs accessed 2026-08-30.

Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. Technical Report RFC 6962, IETF, 2013.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic Harness Engineering: Observability-Driven Automatic Evolution of Coding-Agent Harnesses. arXiv:2604.25850, 2026.

Jiacheng Liu, Xiaohan Zhao, Xinyi Shang, and Zhiqiang Shen. Dive into Claude Code: The Design Space of Today's and Future AI Agent Systems. arXiv:2604.14228, 2026a. Technical report, v2 2 July 2026.

Jiale Liu, Zhiheng Xi, Shaokun Zhang, et al. Who&When Pro: Can LLMs Really Attribute Failures in AI Agents? arXiv:2607.09996, 2026b.

Yi Nian, Aojie Yuan, Haiyue Zhang, et al. Auditable Agents. arXiv:2604.05485, 2026. Short version in Proc. ACM AI Leadership Summit 2026.

OpenTelemetry. Semantic Conventions for GenAI Agent and Framework Spans. `https://github.com/open-telemetry/semantic-conventions-genai`, 2026. Status: Development; accessed 2026-08-30.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM Evaluators Recognize and Favor Their Own Generations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. Practical Whole-System Provenance Capture. In *Proceedings of the 2017 Symposium on Cloud Computing (SoCC)*, pp. 405–418, 2017.

Devin J. Pohly, Stephen McLaughlin, Patrick McDaniel, and Kevin Butler. Hi-Fi: Collecting High-Fidelity Whole-System Provenance. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC)*, pp. 259–268, 2012.

Mateusz Pysklo, Andrii Zhuravel, and Ian Watson. Agent-Diff: Benchmarking LLM Agents on Enterprise API Tasks via Code Execution with State-Diff-Based Evaluation. arXiv:2602.11224, 2026.

Benjamin Rombaut. Inside the Scaffold: A Source-Code Taxonomy of Coding Agent Architectures. arXiv:2604.03515, 2026.

Bruce Schneier and John Kelsey. Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999.

Oleg Solozobov. DEMM-Bench: A Cross-Regime Benchmark for Agent-Runtime Governance-Evidence Sufficiency. arXiv:2606.20634, 2026.

Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. in-toto: Providing Farm-to-Table Guarantees for Bits and Bytes. In *Proceedings of the 28th USENIX Security Symposium*, pp. 1393–1410, 2019.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhaohui Geoffrey Wang. AgentTrace: Causal Graph Tracing for Root Cause Analysis in Deployed Multi-Agent Systems. arXiv:2603.14688, 2026. ICLR 2026 Workshop on Agents in the Wild.

Hu Wei. Architectural Design Decisions in AI Agent Harnesses. arXiv:2604.18071, 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. arXiv:2406.12045, 2024.

Guibin Zhang, Junhao Wang, Junjie Chen, et al. AgenTracer: Who Is Inducing Failure in the LLM Agentic Systems? In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.03312.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *PMLR*, pp. 76583–76599, 2025. Spotlight; arXiv:2505.00212.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Yusheng Zheng, Yanpeng Hu, Tong Yu, and Andi Quinn. AgentSight: System-Level Observability for AI Agents Using eBPF. arXiv:2508.02736, 2025.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, et al. Agent-as-a-Judge: Evaluate Agents with Agents. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *PMLR*, pp. 80569–80611, 2025.

A AUDIT RUBRIC

The static audit of §3 scores each (framework, property) cell on three bands, *present*, *partial* and *absent*, defined property by property below. Ten properties are scored. Five are architectural, drawn from a prior source-level reading of three harnesses (citation withheld for anonymity). Two serve a reader who was not present: a tamper-evident record checkable without trusting the runtime, and domain provenance as first-class fields. Three are *controls*: automated coverage of the control loop, a versioned public surface, a typed configuration schema. A control is a property that ordinary engineering activity produces and that this paper's argument says nothing about; it is scored by the same passes under the same rule, and what the controls showed is in Appendix B. The band text below is what each scoring pass was handed, one property at a time, with one clause added afterwards and noted at the end of this appendix. How the passes were run, checked and settled is Appendix B.

Terms used across the appendices. A *gap cell* is one of the 32 (framework, property) cells on the two absent-reader properties. The *present-serving mean* is a framework's average over the eight properties that serve a reader who was present (present 1, partial ½, absent 0). *Grader 1, 2, 3* are the three models that grade Q2 answers against the task key: `claude-haiku-4-5` (fixed in the preregistration; E1's vendor), `deepseek-v4-pro` (E3's model) and `gpt-5.2`; the headline is their majority. *Witnessed* content is what crossed a boundary the environment-side writer records (model traffic, process executions, file writes), so the ledger holds it independently of the harness; a citation is *independently witnessed* when it lands on such a ledger row or on one of the two benchmark-written anchors. The *like-for-like ledger* of the cross-vendor runs is their ledger cut to the main grid's row kinds, without tool-output rows. The *event reading* (the body's registered rule) and the *narration reading* (its stricter reading) are the two ways to grade a citation that lands on a ledger row holding the agent's own words: the first counts it corroborated for Q1, because the ledger witnessed that the words were said; the second counts it narration, because a witness to the saying is not a witness to the said.

The five architectural properties are banded as follows.

Commoditised loop (architectural). *Present*: the main control loop is small (about a thousand lines or fewer), delegated to a framework, or swappable by configuration, and the project does not market it as its edge. *Partial*: a moderately large custom loop that is still not the selling point. *Absent*: a large proprietary hand-tuned loop positioned as the product's advantage.

Append-only, replayable session record (architectural). *Present*: append-only entries, recovery replays or reads a register, committed history is never mutated. *Partial*: replayable, but recovery rewrites or patches history, for example a checkpoint that edits messages. *Absent*: session state mutable in place, or not replayable.

Model quirks as data (architectural). *Present*: a catalogue, profile or configuration drives quirk handling, so adding a model means adding data. *Partial*: a mix of data and significant inline branching. *Absent*: quirks handled by scattered inline conditionals on the provider.

Progressive disclosure of context (architectural). *Present*: the prompt carries indices (name, description, path) and bodies are read on demand. *Partial*: some on demand and some front-loaded, for example every tool schema always in the prompt. *Absent*: everything front-loaded into the context.

Explicit seams (architectural). *Present*: documented extension points, and a capability can be added without editing the core. *Partial*: some seams, but core edits are usual. *Absent*: a monolith, where extension means forking or patching the core.

The two absent-reader properties are banded next. Their bands were set after a pilot and before the population run.

Tamper-evident record checkable without trusting the runtime (absent reader). *Present*: a hash-chained, signed or attested append-only log that an outsider can check against something the runtime did not write, a second writer or an external anchor; a chain the runtime signs over its own record alone does not qualify (§6). *Partial*: an audit-purposed durable log (dedicated audit events or an exported ledger) whose committed history is preserved append-only, reconstructable only by trusting the runtime that wrote it. *Absent*: no such record, or session persistence for resumption only, or the runtime mutates committed history in normal operation by rewrite in place, fork-drop, update or delete. Tiebreak: truncating an unfinished tail after a crash does not count as mutation; rewriting committed history does, however durable the store.

Domain provenance as first-class fields (absent reader). *Present*: tool results carry and enforce as-of time, entitlement and cost attestation. *Partial*: some domain provenance carried but neither enforced nor complete, meaning a data as-of or validity time, an entitlement or rights tag, or a cost attestation on tool results. Record-creation timestamps, token or currency usage counters and licence metadata do not count, and the provenance must live in the harness-level tool-result schema rather than inside one tool's own payload. *Absent*: none beyond those excluded artefacts.

The three control properties are banded last.

Automated coverage of the control loop (control). *Present*: a test file invokes the loop's entry point and runs at least one complete iteration with the model call substituted by a fake, stub or recorded fixture, under the project's standard test command. *Partial*: tests cover components the loop calls (tool dispatch, prompt assembly, response parsing, session writes) but none drives the loop itself, or a loop-level test exists but needs a live model, network access or an API key. *Absent*: no automated test exercises the loop or its immediate components, or the only end-to-end path is a benchmark runner against a live model.

Versioned public surface (control). *Present*: a declared public surface (package entry point, plugin or extension API, or an interface the documentation marks stable) carrying an explicit version, and at least one in-code compatibility mechanism that keeps an older form working across a change. *Partial*: a declared and versioned surface with nothing in code preserving an older form, or isolated deprecation markers with no declared surface behind them. *Absent*: no declared public surface, the version string is a release number only, and no compatibility or deprecation machinery anywhere in the tree.

Typed configuration schema (control). *Present*: user-facing configuration keys are declared in an explicit schema, validation runs at load, and invalid or unknown input produces a diagnostic naming the offending key or path. *Partial*: configuration is parsed into typed structures or partly validated, but unknown or invalid keys pass silently or fail only where the value is used, or a schema covers part of the surface only. *Absent*: configuration is read as an untyped map and consumed ad hoc, and the set of valid keys is discoverable only by reading the consuming code.

The clause in the tamper-evident band that a chain the runtime signs over its own record alone does not qualify was added on 2026-09-19, after both instruments had scored; neither instrument's passes saw it and no cell was rescored under it. It writes down the reading the passes had already applied: no cell was scored `present` on such a chain under either instrument, and the one cell where a single pass read one as such, aider's git auto-commits under the second instrument (Appendix B), stands `absent` in the reported grid.

The full rubric text, the control adjudication rules and the re-scan protocol ship with the released audit grid.

B AUDIT PROCEDURE AND CHECKS

Every cell of §3 was scored by three blinded model passes, checked mechanically, and re-checked four ways. No check moves a *gap cell*, one of the 32 cells on the two external-verifiability properties, to `present`. The four checks are a completeness critic on every absence and a human re-judging of 33 cells. They also include the model the preregistration named (the *registered instrument*, `claude-fable-5`) on nine frameworks, and a second instrument from another vendor (`deepseek-v4-pro`) on every cell. The grid of §3, scored by `claude-sonnet-5`, is called the *reported grid* throughout, and `claude-sonnet-5` is called the instrument. This appendix

gives the population, then the procedure and its checks, then what the controls showed, then how to reproduce the checks. Table 3 collects what each check compared and what it returned; the paragraphs below define its terms. A cell is named by framework and property number, P1 to P10 in the order of Appendix A. P6 is the tamper-evident record and P7 domain provenance; these are the two absent-reader properties.

Table 3: The checks on the audit grid of §3, what each compared, and what it returned: none moves a gap cell to `present`. Detail in the paragraphs of this appendix and in Appendix H, rows B1 and B9.

check	what was compared	result
three passes per cell	agreement among the three passes, 160 cells (16 frameworks × 10 properties) at the pinned commits	unanimous on 128; on 27 of the 32 absent-reader cells
completeness critic	every `absent` cell re-read by a different model of the same vendor (Claude Opus), misses re-scanned by three fresh passes	46 cells named; 5 flip, none to `present` on an absent-reader property
human re-judging	two annotators on 33 other cells (controls scored `present`, architectural properties scored `absent`), blind to the instrument	26 of 33 agree (78.8%, Wilson 95% CI [62, 89]), under the registered 80%; no gap cell in the sample
registered instrument	`claude-fable-5` against `claude-sonnet-5` on 63 cells of nine frameworks	48 exact, 60 within one band; all 18 gap cells agree
activity	Spearman's ρ, present-serving mean against commits, sixteen frameworks	mean 80% (0.80 on the 1, ½, 0 scale over the eight properties); $\rho = 0.55$, two-sided permutation p = 0.03 (50,000 shuffles); no absent-reader cell `present` at any activity level
second instrument, second vendor	`deepseek-v4-pro`, three passes per cell over all 160 cells, same rubric and commits, GPT-5.2 critic; registered before the run, 2026-09-19	129 exact, 159 within one band; 31 of 32 gap cells agree, none `present`; re-judging of the 31 disagreements sides with the reported grid on 18, aider P6 ruled `absent`

Sixteen frameworks, each read at a pinned commit. The sixteen are the thirteen harnesses of the source-code taxonomy of Rombaut (2026), taken as that survey's population, plus the three of the prior source-level reading behind Appendix A. A harness without public source at a pinnable commit is excluded, which is why Claude Code is not among them. The SWE-agent family (SWE-agent, mini-swe-agent, DARS-Agent) is kept whole rather than collapsed to one member. Table 4 gives the commit each framework was scored at, the date of that commit, and its position on the activity axis of Fig. 6.

Activity, not age, tracks the present-serving mean. The activity axis counts commits on the default branch between 2025-08-25 and 2026-08-25 inclusive. The count was read in the clones as they stood on 2026-08-25. The *present-serving mean* is a framework's mean band over the eight properties that serve a present reader. Across the sixteen pinned commits, Spearman's ρ between commit count and the present-serving mean is 0.55. The permutation p is 0.03 over 50,000 shuffles. Three other proxies, read on the same date, give:

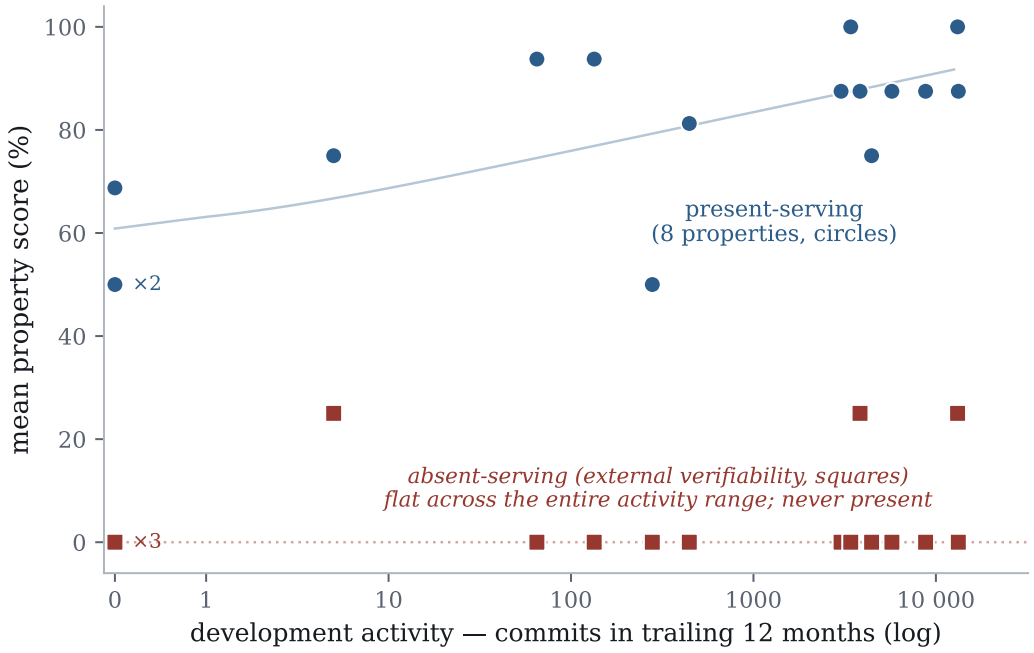


Figure 6: Present-serving properties (circles, mean over eight) and external verifiability (squares, mean over two) for the sixteen frameworks against their commit count in the year to 2026-08-25 (log axis, zero on a linear stub). The line is a least-squares fit on log(commits + 1) to the circles only, sixteen frameworks; the dotted line is the zero reference; $\times n$ marks points that coincide.

Table 4: The sixteen frameworks of §3 with the commit each was scored at, the date of that commit, and its position on the activity axis of Fig. 6 (commits on the default branch between 2025-08-25 and 2026-08-25). The five starred are the harnesses of §4, run at these same commits.

framework	repository	commit	commit date	commits
opencode*	`sst/opencode`	`105b398c2`	2026-08-24	13274
gemini-cli	`google-gemini/gemini-cli`	`5411f113c`	2026-08-21	4441
codex	`openai/codex`	`e3609f2d0`	2026-08-24	8779
OpenHands	`All-Hands-AI/OpenHands`	`8511fff62`	2026-08-24	3021
cline*	`cline/cline`	`09ee90263`	2026-08-23	3838
aider*	`Aider-AI/aider`	`5dc9490bb`	2026-05-22	134
SWE-agent*	`SWE-agent/SWE-agent`	`3ea751c08`	2026-07-16	65
mini-swe-agent*	`SWE-agent/mini-swe-agent`	`25941c89c`	2026-08-17	445
auto-code-rover	`nus-apr/auto-code-rover`	`585d3e639`	2025-04-24	0
Agentless	`OpenAutoCoder/Agentless`	`5ce5888b9`	2024-12-22	0
moatless-tools	`aorwall/moatless-tools`	`011ead57a`	2025-09-01	5
Prometheus	`EuniAI/Prometheus`	`acb83608e`	2026-08-16	279
DARS-Agent	`darsagent/DARS-Agent`	`eab35168a`	2025-05-20	0
deepagents	`langchain-ai/deepagents`	`2c8015378`	2026-08-23	3413
pi	`earendil-works/pi`	`a470b121b`	2026-08-23	5737
deepseek-harness	`deepseek-ai/deepseek-harness`	`b150a551b`	2026-08-21	13147

- all-time contributor count: 0.50
- release-tag count: 0.49
- repository age, first commit to 2026-08-25: -0.20

The youngest projects are the most active, so the axis measures activity, not age. The absent-reader zero also holds back in time. Each framework was scored at its pinned commit and up to four earlier commits, 54 framework-time points in all (Appendix H, row B1). The 22 of these that carried all ten properties give 44 absent-reader cells, and none of them is `present`. §3 reads the pinned commits.

The location index tells a pass where to look, never what to find. Before any scoring, a separate model pass reads each repository and writes a *location index*. It lists the paths and search terms where each property's mechanism would live if it existed: the loop, the session store, the quirk layer, prompt assembly, the extension mechanism, telemetry. It names paths and search terms and nothing else. The index is audited before use and rejected if it carries evaluative or architectural vocabulary, counts of lines or files, or any claim that a mechanism lives outside the tree. That index points at none of the places a control lives, so a supplementary scouting pass writes a second index per repository. The second index covers tests and test-runner configuration, the package manifest and exported entry points, and configuration loading. It carries the loop and session entries over verbatim, so the control on loop coverage is not handicapped. No earlier pass is re-run against it.

A pass sees one property and returns located evidence. A pass is handed one property's bands (Appendix A), the evidence rule below, and the location index of the repository it is reading. It is never told the other properties, the claim of §3, or an expected answer, so it cannot score toward the claim. It returns one JSON object and nothing else: the property, the score, an evidence list, a counter-evidence list, a rationale of at most three sentences, and a confidence. Each evidence item names a repository-relative path, a line range, a verbatim quote of at most 160 characters, and one clause saying what the quote shows. Where an absent-reader property is scored `absent`, the evidence is the files checked where the mechanism would live, quoted for what is there instead. The audited unit is the product whose repository Table 4 names. Where a repository publishes more than one product, only that product's subtree is in scope. Evidence from a sibling product does not support a score.

A score without a real file and line is discarded. A score whose evidence list holds no real in-repository file and line is invalid. A committed checker re-reads every archived pass after each batch. It confirms three passes per cell, the frozen model and rubric-version fields, and the property field against the file name. It also confirms that every cited path exists in the checkout at the pinned commit. For deepagents, the one repository that publishes more than one product, it confirms that every path lies inside the named product's subtree.

Three passes per cell, and the majority sets the verdict. One model, `claude-sonnet-5`, scores every framework and every property. Each archived pass records the model and rubric version it ran under. Three passes score each cell, and the majority sets the verdict. When the three split two to one, the majority still stands. The split stays in the data with its three symbols, confidences and evidence count, and no fourth pass is run. Six of Prometheus's cells were scored the same way in a separate run under the instrument's own location index.

A critic re-reads every absence. A completeness critic reads every cell scored `absent` and answers one question. Is this a true absence, or an unread area? A verdict of `named_miss` must name the files that would have to be checked. A named file that is not in the checkout is dropped. The critic runs on a different model of the same vendor as the instrument (Claude Opus against Claude Sonnet), so the second read is not the first read repeated.

Re-scanning what the critic named moves no absent-reader cell to `present`. The critic reads the 77 `absent` cells of the five audit time points (41 at the pinned commits). It names 46 of them (30 at the pinned commits). Each is re-scanned by three fresh passes whose location index carries the critic's named files. Five flip, none to `present` on an absent-reader property. One flips an absent-reader cell from `absent` to `partial`, the third of the three partials §3 reports. The thirty-two absent-reader cells settle at zero `present`, three `partial` and twenty-nine `absent`. Every `partial` is an operator-facing audit slice, reconstructable only by trusting the runtime that wrote it.

A second instrument from a second vendor also finds no gap cell `present`. Every cell of the grid was scored again by `deepseek-v4-pro`, and no gap cell came back `present`. It ran as a tool-using pass over the same blobless checkouts at the pinned commits. Each cell got three blinded passes under the same rubric text and evidence contract. Citations were checked against the checkout under the rule the reported grid was held to. Its completeness critic was `gpt-5.2` through the Batch API, handed each repository's file listing in place of a shell. It named every DeepSeek-

`absent` cell, 40 in all. The 120 fresh re-scans flipped three majorities, none to `present` on a gap cell. Aider P6 and moatless-tools P6 moved from `absent` to `partial`, and codex P1 from `absent` to `present`.

The two instruments agree on most cells, and on every gap cell within one band. The cross-vendor majorities match the reported grid's majorities exactly on 129 of 160 cells, 80.6%. Within one band they match on 159. On the 32 external-verifiability cells they match on 31 exactly and on all 32 within one band. None of these cells is `present` under either instrument. The activity correlation carries over as well. Under DeepSeek the present-serving mean correlates with commit count at $\rho = 0.55$, the reported grid's value to two decimals. The two-sided permutation p is 0.03 for both grids, and the one-sided p reads 0.014 for both. Both use 50,000 shuffles over n = 16.

Under the second instrument the gap column holds 28 `absent` and four `partial`. The four are the three the reported grid marks `partial`, plus aider P6. On aider P6 the three re-scan passes split one `absent`, one `partial` and one `present`. The `present` vote read aider's git auto-commits as a chain. The pass rubric carried the band text the reported grid's passes saw, not the clause added to it on 2026-09-19.

Three of the four registered predictions held; the fourth failed on its count. We registered four predictions before the run:

- P38.1 held: at least 30 of the 32 gap cells agree, and none is `present`.
- P38.2 held: exact agreement is at least 75%, and agreement within one band at least 90%.
- P38.3 held: the activity correlation stays positive.
- P38.4 failed: it said the critic would name fewer than 15 gap cells.

The critic named all 40 `absent` cells, 30 of them gap cells. The re-scans then moved two of the 30 to `partial`, leaving the gap column at 28 `absent` and 4 `partial`. The reported grid stays the reported grid, and its 31 disagreements with DeepSeek go to blind human re-judging. The reported grid is higher on 19 of them and DeepSeek on 12. Only one cell, DARS-Agent P2, is two bands apart.

Human re-judging tests the instrument where nothing is at stake, not the zero. Two annotators of the §5.4 panel re-judged 33 cells blind: every cell where the instrument scored a control property `present` (28: loop-test coverage 12, versioned public surface 11, typed configuration 5) and every cell of the three-pass grid where it scored an architectural property `absent` (5; Prometheus's model-quirks cell, scored in the separate index run, was not sampled). The sample holds no gap cell. The annotators saw the rubric bands and the evidence contract, not the instrument's score or rationale. Where the two split, the cell counts as `partial`, the registered tie rule. Against the instrument, 26 of 33 agree (78.8%, Wilson 95% CI [62, 89]): 23 of the 28 control cells and 3 of the 5 architectural cells. That is under the preregistered floor of 80%, so by the reading registered in advance (Appendix H) §3 states the activity pattern as the instrument's; the interval includes 80%, so the miss is not clear-cut either. Read one at a time, which the preregistration did not specify, each annotator agrees with the instrument more often, 29 of 33 (87.9% [72.7, 95.2]) and 28 of 33 (84.8% [69.1, 93.3]); the pair rule lowers the count because each of the 7 splits scores `partial` and none of those cells is `partial` in the grid. All seven disagreements are splits between the two annotators. On five control cells the instrument's `present` becomes `partial`: cline and moatless-tools on versioned public surface; SWE-agent, codex and auto-code-rover on loop-test coverage. On two architectural cells its `absent` becomes `partial`: DARS-Agent's commoditised loop and Prometheus's append-only session record.

Human re-judging of the disagreements. Two other annotators read the 31 cells where the two instruments disagree, with both instruments' evidence in random order and no instrument named, and chose a band per cell; a split counts as siding with neither. They sided with the reported grid on 18, with the second instrument on 9, and with neither on 4; agreement with the reported grid is 58% (95% CI [41, 74]). On the one external-verifiability cell in dispute, aider P6, both sided with the reported grid and ruled `absent`, reading the git auto-commits as session persistence, not a chain an outsider could check; the count of 0 present / 3 partial / 29 absent stands. Of the three cells the critic's re-scans flipped, two were in dispute: aider P6, as above, and codex P1, where the annotators

split, one siding with the reported grid's `partial` and one with neither; moatless-tools P6 agreed between the instruments and was not re-judged.

The registered instrument agrees with the reported grid on every gap cell both scored. `claude-fable-5`, the model the preregistration had named as the instrument, scored nine of the sixteen frameworks. Both it and `claude-sonnet-5` were registered as measurement sets, with neither primary. The `claude-sonnet-5` set was designated the reported one on 2026-09-09, because it alone covers every framework and time point. The `claude-fable-5` set is kept as a replication of the registered instrument. Neither set was extended afterwards (Appendix H, row B9). Where both cover a cell they agree as Table 3 states, and neither marks a gap cell `present`.

The controls show the zero is not the mark of a strict instrument. The three control properties check the instrument itself. They are properties any active project acquires, chosen before any control cell was scored and scored by the same passes under the same rule. They come back filled: 28 of the 48 control cells are `present` at the pinned commits. No control sits materially below the five architectural properties across the population. §3 reports this as the instrument not separating architecture from ordinary engineering activity. Two caveats apply. We wrote the control rubrics after the architectural results were in, which if anything favours controls that behave differently from the architectural properties; they did not. And the architectural properties were scored before the controls existed, so the controls could not have shaped those scores. So the instrument reaches `present` readily where nothing is at stake, and the zero across the thirty-two absent-reader cells is an absence.

On a finer ladder, no framework reaches a second writer (preregistration Addendum 42). A single band cannot show how far short a framework falls. So after scoring, each framework was scored once more on a five-rung ladder, by the same instrument, tools and evidence rule, three blinded passes each. The rubric and predictions were registered before the first pass. The rungs are:

- rung 0: no durable record, or one the runtime rewrites or deletes in normal operation;
- rung 1: a single-writer append-only record, a checksum or chain the runtime computes over it included;
- rung 2: a second writer outside the runtime, reconciled;
- rung 3: rung 2 with an externally anchored head;
- rung 4: a multi-party ledger.

All 48 passes carried located evidence. Fourteen frameworks sit at rung 0 and two at rung 1; one of the two, deepseek-harness, has a checksum it computes over its own record. None reaches rung 2, as predicted before the first pass (P42.1: none at rung 2; P42.3: at most three with a self-computed checksum). Prediction P42.2 held only in part. It said all three frameworks partial on the tamper-evident property sit at rung 1, but only deepseek-harness does. The reason is that the band scores the audit log and the ladder the session record. Cline and moatless-tools each keep an append-only audit log (`hooks.jsonl`, `events.jsonl`), which makes them partial on the property. They also rewrite or delete their session records in normal operation, which puts them at rung 0. Aider's passes split 0, 1, 1. Its chat history is only ever appended, but it is an ordinary file, and the rubric left open whether that is rung 0 or 1.

Table 5: The record-integrity ladder at the pinned commits: median of three passes per framework (Addendum 42).

rung	frameworks	passes
0	Agentless, auto-code-rover, cline, codex, DARS-Agent, deepagents, gemini-cli, mini-swe-agent, moatless-tools, OpenHands, opencode, pi, Prometheus, SWE-agent	0, 0, 0 each
1	aider; deepseek-harness (self-computed checksum)	0, 1, 1; 1, 1, 1
2–4	none	—

Reproduction. Four scripts reproduce the second-instrument check. A *pinned commit* is the commit a framework was scored at (Table 4). A *blobless checkout* is a clone with full commit history whose

file contents are fetched only for the commit checked out. A *majority* is the band at least two of a cell's three passes returned.

- `ds_pass.py` scores one pass: a blinded, tool-using read of one framework for one property over its blobless checkout at the pinned commit. It uses no location index and saves the pass as one JSON file. It keeps a pass only if the pass meets the citation rule the reported grid was held to: every cited path exists in the checkout, inside the product's subtree.
- `build_grid_deepseek.py` turns each cell's three passes into a majority and writes the cross-vendor grid.
- `critic_gpt.py` runs the *critic*, a second model that reads every `absent` cell and names the files a miss would live in. `ds_pass.py --rescan` then *re-scans* each named cell with three fresh passes handed those files, and the grid is rebuilt.
- `deepseek_agreement.py` compares that grid with the reported grid (`scale/master-grid.json`) and writes the agreement file and the blind disagreement sheet. The critic's inputs and verdicts sit under `scale/critic-input-deepseek/`. Its counts: 129 of 160 cells agree exactly, 31 of the 32 gap cells agree, and no gap cell is `present`.

The reproduction target is the zero on `present`, not the 129, because passes are stochastic and a re-run yields its own passes and counts. The archived passes are the record. The reported grid's passes were agent sessions run from the rubric and each framework's location index, not a script. They are archived as `scale/evidence/<framework>/E<k>_run<n>.json` (property k), and `build_grid.py` and `verify_cells.py` rebuild the grid from them.

Sources. The script and results file behind each paragraph of this appendix are listed, by paragraph, in the release (`RELEASE-README.md`, "Appendix sources").

C TASK SET AND GROUND TRUTH

The task set of §4.1 holds fourteen tasks: eight prototypes and six SWE-bench Verified derivations. The eight prototypes re-implement production failure mechanisms in fresh minimal repositories with synthetic data.

The six SWE-bench tasks are the lowest eligible issue id in each of three repositories, plus the next lowest in each. They were picked by a content-blind rule, frozen before any run, so that no task could be chosen for what its mechanism looks like (Appendix H, entry of 2026-08-28). The rule draws on the instances of SWE-bench Verified annotated 15 minutes to 1 hour in the pure-Python repositories. It keeps the three repositories with the most eligible instances (django, sphinx, pytest), and within each it takes the lowest numeric issue id. A pre-registered substitution queue, in the same order, replaces any instance that fails the entry check. The three added after registration are the next lowest eligible id in each repository, the head of each queue. In sphinx the first two failed the entry check and were replaced by the third, 8035.

Every task passes four gates: the entry test fails at base, passes with the reference fix, fails again on revert, and behaves identically without network access. Each key also lists the facts that establish the mechanism without relying on anything the agent said.

Accuracy grading (§5.1) covers all fourteen tasks, and dropping the three starred tasks barely moves it. Their keys were written after the examinations, from the upstream gold patches, so for those tasks the answers precede the keys; each key's provenance header says so. Without those three tasks, native attribution under grader 1 is 99% for E1 (109 of 110), against 97% on all fourteen. For E2 it is 78% (86 of 110), against 76%. No condition moves by more than 4.4 points under grader 1 or grader 2. The E1 to E2 gap under native goes from 21.4 to 20.9 points (`bench/exam/KEYS-SENSITIVITY.md`).

task	type	mechanism (one line)
attribution-from-query-parameter	prototype	`--scope` is only a probe hint; `index.find` tries the named batch first
default-view-drops-assurance-tier	prototype	consumer-facing read path projects away the assurance tier
dry-run-flag-that-writes	prototype	the dry-run branch still writes
evidence-attached-after-decision	prototype	decision and evidence read as two separate passes
merge-key-rebinds-to-next-period	prototype	rebind matches another row's stored label, not temporal adjacency
success-reported-for-work-not-done	prototype	success recorded for work never performed
upstream-failure-swallowed-to-benign-empty	prototype	per-batch failure swallowed into a benign stub envelope
watermark-advances-on-nonatomic-load	prototype	progress line computed over a non-atomic load
swebench-pytest-5631	SWE-bench	`num_mock_patch_args` miscounts `@mock.patch` decorations
swebench-django-13809	SWE-bench	`squashmigrations --squashed-name` silently ignored
swebench-sphinx-7748	SWE-bench	`_find_signature` matches only the first signature line; later overloads are dropped
swebench-django-13925*	SWE-bench	W042 misfires on inherited primary keys (`_check_default_pk` lacks the parent-link exclusion)
swebench-pytest-5840*	SWE-bench	`unique_path` lower-cases conftest paths; cache keyed by the normalised path
swebench-sphinx-8035*	SWE-bench	`:private-members:` typed `bool_option`, cannot take member names

D MECHANICAL EXTRACTION AND FOLDING

Two mechanical steps turn each harness's native record into the examiner *bundles* of §4.3, a bundle being the set of files one condition hands the examiner (Table 6). The first, extraction, runs on every record. It removes only packaging: repeated copies of identical content, binary containers, and task infrastructure picked up by the collection rule, which takes every file under the run's home and workspace. For mini-swe-agent and SWE-agent, whose records repeat themselves, that de-duplication happens at extraction for every bundle. Extraction never removes content that appears once. The second step, folding, runs only when a bundle does not fit an examiner's context. It replaces repeated content with a placeholder carrying a content hash. Unlike extraction, folding can elide content that appears once, and it records what it elided. The folding factor is reported as a result.

The table is the condition. The assembler inlines every file in the bundle directory, so Table 6 lists exactly what each condition hands the examiner. Ceiling carries the same three kinds of material as native+ledger, plus the derived decision list. On ten of the 140, its record and ledger are folded further than the native copies, because those bundles were over the window. Ceiling still lacks the operator stream, so it is not a superset of trace.

Four properties hold the extractor to account. Extraction is held to no discretion, determinism, auditability and uniformity, so that anyone can rebuild the bundles and check them. No discretion: the extractor addresses fixed paths, fixed table names and content fingerprints, and holds no judgement about what matters. Determinism: a re-run over the frozen corpus produces byte-identical output. Every file written is recorded with its length and its SHA-256 in a per-bundle manifest. Auditability: the original artefacts travel with the release, and their hashes tie to the corpus index. A reader can therefore re-run the extractor and reconcile. Uniformity: one script covers the five harnesses. Per harness it knows only the path the record is written to and the one fixed operation for that format, below.

Each harness's record gets one fixed operation. Every path named here is inside the run's isolated home or workspace. **aider** leaves `.aider.chat.history.md` and

Table 6: What each condition's bundle contains, as built: every condition above floor carries the two anchors, and apart from the anchors only the ledger and the external capture are written outside the harness. The two benchmark-written anchors are the entry test's failing output at base and the final workspace diff. Every bundle that carries them also carries the workspace file-tree listing.

condition	what the examiner is handed
floor	the workspace file-tree listing, and the final contents of the changed workspace files that match the source-extension whitelist. Where none matched, the tree listing alone (23 of 140); no anchors
trace	`external-trace.tsv`, the sandbox's process and file events (the same events the environment-side writer adds to the ledger), and `operator-stdout.log`, the harness's terminal output as the sandbox captured it (the *operator stream*); the two anchors
native	`record/` as extracted; the two anchors
native+ledger	native, and `ledger/`: `events.tsv`, `chain-head.txt`, `reconcile.txt`, and a README stating the folding rules; the two anchors
ceiling	`record/`, `ledger/`, and `ground_truth_decisions.txt`, the mechanically derived decision list. Ten of the 140 carry a record and a ledger folded further than the native+ledger copies; the two anchors

`.aider.input.history` as dotfiles in the workspace. Both are already text and are copied unchanged. **cline** writes the session JSON and the messages JSON under the session directory, also copied unchanged. **opencode** writes `opencode.db`, a SQLite database with a write-ahead log. Extraction merges that log, then dumps the `event`, `message`, `part` and `session` tables as text in that fixed order, dropping nothing. **mini-swe-agent** writes `last_mini_run.traj.json`, and **SWE-agent** writes the run's `.traj` file under `trajectories/`. Each becomes a JSON view with repeated blocks of 256 bytes or more replaced by a fingerprint.

The two trajectory formats are de-duplicated at extraction, always. Both re-embed the whole conversation at every step. This is the only place extraction removes anything from the record itself, and it removes only blocks already in the bundle. A string of 256 bytes or more that has already appeared is replaced by a placeholder carrying the first sixteen hex digits of its SHA-256. The first occurrence is kept in full. This is rule N1 of the folding below, applied to these two formats unconditionally, whether or not the bundle fits. Shorter strings, such as key names and short commands, are kept as written. The threshold is frozen and ships with the script.

Extraction removes packaging, never content that appears once. Removal is confined to three kinds of packaging. First, repeated copies of content already in the bundle. Second, binary containers, such as one harness's snapshot object store, which is listed as a directory instead. Third, task infrastructure picked up by the collection rule: the language toolchain, the virtual environment, the package cache and the model-capability caches. The workspace source tree is listed rather than inlined, because a SWE-bench repository is reference material and not part of the record. Every bundle carries a manifest naming what was excluded, the length and hash of every file written, and the source path each file came from. Every bundle that carries a record also gets three files: the entry test's failing output before the run, the final workspace diff, and the workspace file-tree listing. The first two are the benchmark-written anchors of §4.2, written by neither the harness nor the examiner.

Folding runs only when a bundle would not fit. A bundle over one million estimated tokens is folded. Tokens are estimated as bytes / 3.5 $\times$ 1.95. The factor of 1.95 corrects an undercount: on four bundles, bytes / 3.5 alone undercounted the examiner's own token count by a factor of 1.85 to 2.08. Five mechanical rules do the folding: R1 to R3 on the ledger as presented, and N1 and N2 on the native record.

- **R1** replaces an environment-side body in a ledger event that already appears verbatim in the same bundle, including under JSON escaping. In its place goes a placeholder carrying the body's fingerprint, so the reader finds the original in the material.
- **R2** shortens any hash column in the presentation copy of the ledger to its first sixteen hex digits, with the database as the authority for the full value.
- **R3** replaces twenty or more consecutive ledger events of the same writer and kind with empty bodies by one summary row. That row names the sequence range, the count, the kind, and a fingerprint of the block.
- **N1** replaces a line of 256 bytes or more in a record file that has already appeared in that file with a placeholder carrying the line's fingerprint.
- **N2** keeps the first and last 8 KB of a record file over 256 KB and replaces the middle with a placeholder naming the byte count and a fingerprint.

Ceiling bundles still over the window after these rules received one further pass at tighter thresholds. That pass applies to the record and the ledger alike: 128-byte lines, 64 KB files, and 4 KB of head and tail.

Every elision can be checked. Every placeholder names what it replaces and carries a fingerprint of it, so a reader holding the released originals can verify it. The ledger in the database is never folded, and the chain is verified against the database rather than against the presentation copy. The anchors and the derived decision list are never folded either. Placeholders do have a cost: they explain part of what E2 flags on clean copies. On SWE-agent, whose records are full of de-duplication placeholders, E2 flags 18 of the 20 clean copies. Across all pairs, placeholders raise its false-positive rate by about ten points (Table 25).

Folding is recorded per bundle and reported. The ledger manifest's `fold` field records the rows before and after, the number of units folded, and the bytes on each side. The material manifest's `elision` field records the same per file. Before folding, twenty bundles exceed the window. After it, none does on its record alone: the largest is under 400k tokens, and the one that had reached 1.54M falls to 105k. Eight E1 and seven E2 ceiling bundles still exceed the window once their decision lists are added, and are excluded (App. F). Folding shows up twice in the results. The first is in what E2 flags on clean records (§5.3). The second is in 15 of the 19 clean-copy mismatches the reconciliation query reports. The other four are tool outputs cline cut to 8,000 characters before sending them to the model. There the witnessed unit is the cut form, and the record holds the full one (Appendix G).

The full extraction and folding rule text ships with the released pipeline.

E MECHANICAL CITATION SCORING, PER RECORD FORMAT

A citation is graded by where its excerpt lands, never by the examiner's own label for it. This section gives the landing-site rules that grade every citation of §4.3, and the adversarial gate that the Q3 injection of §5.3 had to pass. A citation's *landing site* is the file, and the row or field in it, that holds the cited excerpt. The *fragment matcher* is the routine that finds the excerpt there: verbatim, verbatim under JSON escaping, or as ordered fragments where the examiner's quote has an ellipsis.

The body's numbers come from two pipelines, listed with their results files in the table below. Citation scoring is `bench/exam/score_citations.py`, which `score_grid.py` drives over every examiner answer file; it writes `GRID-SCORES.json`, the source of the corroborated shares of Table 1. The alteration gate starts from `bench/tamper/inject.py`, which writes the clean and the altered copy of each record. Then `redteam.py` shows each pair to the gate model and writes `redteam-window.json`, and `redteam_deepseek.py` repeats that with the second judge. They give 57.5% for the first gate and 60.9% for the second (§5.3).

Each citation gets two scores. The first is verifiability: the excerpt must appear in the cited file, verbatim, verbatim under JSON escaping, or as ordered fragments split on ellipses. A citation that passes is *located* ("located verbatim" in Table 1). One that fails is unsupported and counts against the examiner's verifiability figure. The second is evidence grade, a function of the landing site alone.

The examiner's self-label is recorded but never used, so an examiner cannot raise its own grade. The landing rules, in full:

landing site	rule
`anchors_*`	corroborated (the pre-run failure output and the final workspace diff)
ledger `events.tsv`, row kind `exec` / `file_write`	corroborated
ledger `events.tsv`, kind `model_text` / `tool_input`	corroborated for Q1 (the ledger witnesses the occurrence); narration for Q2 (a witness to the saying is not a witness to the said)
ledger `events.tsv`, kind `tool_result`	not covered by the rules as fixed (the tool output as fed to the model, a row kind added on 2026-08-30, after the main-grid bundles were extracted: no main-grid examiner saw it (Appendix G), Appendix G): narration under the primary rule, witnessed in the sensitivity of Appendix G
ledger row the fragment matcher cannot locate	unlocatable, never corroborated (tightened after examination; below)
ledger reconcile / chain-head files	corroborated (audit-side output)
trace: `external-trace.tsv`	the env-side stream, same format as the ledger; same kind rules as `events.tsv` (added after registration; Appendix H)
trace: `operator-stdout.log`	the externally captured operator stream; corroborated for Q1, narration for Q2 (scoring it fully corroborated shifts trace by +3.9 points; Table 28, the table of every alternative reading, Appendix F)
record, sweagent `.traj`	`observation` fields corroborated; `thought`/`response`/`action`/`query` narration
record, mini traj JSON	roles `tool`/`user`/`system` and `raw_output` corroborated; `assistant` narration
record, cline messages JSON	role $\neq$ assistant corroborated; assistant narration
record, aider markdown	lines opening > corroborated, else narration (in the released records those lines are aider's status, prompt, and echo lines, never command output; sensitivity below)
record, opencode dump	event-table rows corroborated; assistant-marked message rows narration

An excerpt with its grades. Two lines from one mini-swe-agent record (`default-view-drops-assurance-tier`, attempt a02) show the rules at work. The tool message reports that a test file was unchanged, and the next assistant message restates it as "All checks pass". The first lands as corroborated, the second as narration.

```
"raw_output": "test_report.py unchanged confirmed... [corroborated]
"content": "All checks pass and `test_report.py` was not... [narration]
```

Two model readers mostly agree with the rules. Two models were shown the same 100 sampled citations: the grader's (claude-haiku-4-5, E1's vendor) and E3's (deepseek-v4-pro). They agreed with the rules 72% and 68% of the time, and with each other on 90 of 100. Where they disagreed with the rules, the disagreement was symmetric in direction. A 100-citation human verification list ships with the data; human annotators rated the same 100, single-rater on 90 and by majority on the ten shared (results in the human-panel paragraph of Appendix F).

Two rules were tightened after examination, and both lowered a share. The unlocatable rule never corroborates a ledger row the fragment matcher cannot locate. It took E1's pooled corroborated share over native and native+ledger from 73% to 71.4%. E2's fell from 42% to 40.8%. The originals of the re-scored answers are archived. The operator-stream rule grades that log corroborated for Q1 only: the sandbox witnessed that the harness printed the lines, but the harness wrote them. It lowered trace by 3.9 points.

A strict aider rule only strengthens the headlines. No aider run executed a shell command. All 28 records are single-turn, and each "Run shell command?" prompt is answered n. So every one of the 685 lines opening with > is a status line, an echoed prompt, a filename or declined command, or the model's own diff fence, never command output. Scoring those lines as narration flips 411 citations, all in aider. It moves every headline in the direction of the paper's claims and reverses none:

corroborated share (95% bootstrap over runs, B = 10,000, on the E1 rows)	registered rule	strict rule
E1 native	68.8%	63.1% [59.2, 66.9]
E1 native+ledger	74.1%	70.6% [67.6, 73.6]
E2 native	41.1%	35.1%
E2 native+ledger	40.6%	37.1%
E3 native	48.4%	45.1%
E3 native+ledger	46.5%	43.7%
aider, E1	66.2%	25.8%
aider, E2	67.1%	17.1%

E1's ledger shift grows from +5.3 to +7.5, with bootstrap interval [+3.5, +11.6]. E2's stays not significant (+1.9). The E1 to E2 gap is 28.0 points. The self-report gaps widen to 24, 61, and 49 points. The haiku validator agrees with the registered rules on 72 of 100. It sides with the strict rule on the status lines but reads the echoed prompts as terminal transcript, so under the strict rule it agrees on 69 of 100. The registered rule stays primary because it was the convention fixed before scoring, not because the validator separates the two.

The Q3 injection had to pass an adversarial gate before any examination. E1's own model, claude-sonnet-5, was shown both copies of a record side by side and asked which was edited. The injector was repaired until the gate could not pick the edited copy significantly above chance; a bound of 65% was also set before the run. After five repair rounds, each a fix of the injector and a re-run, it distinguished them at 57.5% (65 of 113). That is not significantly above chance (one-sided binomial p = 0.066). Before the first repair it stood at 83.7%, on 82 of 98 pairs whose prompts then fit the window. By round, accuracy went from 83.7 to 70.9 and then 64.5%. It rose to 70.3% in round four and ended at 57.5%. The rounds repaired injection artefacts, not the alterations themselves. They fixed cloned entries without fresh identifiers and timestamps, and deleted messages that left token counts unadjusted. They also fixed rewrites that broke aider's code fences or placed a passing verdict outside an output position.

The pass does not hang on the two unparsed answers. The gate ran against all 115 pairs. Two of the 115 fifth-round answer sheets are not counted, so the denominator is 113. One mixed prose with JSON and parsed to a wrong answer; the other hit the output-token limit with an empty body. Counting both as wrong gives 65 of 115, p = 0.096. Counting both as correct, the reading least favourable to the gate, gives 67 of 115, p = 0.046, still under the bound.

The gate is tuned to E1's family. Because the gate model is E1's, the five rounds repaired what E1's family sees. E1's detection and false-positive rates in Table 1 are conditioned on that. An equal-length rewrite of a test verdict leaves the traceback and exit code in place. That class is therefore easier than an adversary would make it, and its detection rate is an upper bound.

Under a second judge, the gate falls between its registered readings. The same 115 pairs, window, prompt and A/B assignment were judged once each by deepseek-v4-pro (thinking disabled, direct API), with no further repair. It distinguished them on 70 of 115 pairs. That is 60.9% [51.7, 69.3], with one-sided binomial p = 0.012. The entry registering this run fixed two readings in advance, each with a significance condition as well as an accuracy one. Below 65% with $p \geq 0.05$, the gate holds for a second vendor. At 65% or above with $p < 0.05$, it held only against E1's family. The result falls between them and is reported as such: for this judge the gate did not hold, though accuracy stayed within the registered bound. The signal is in the fabrications and in opencode. The judge told apart 33 of 41 fabrications (80%), where the sonnet gate told apart 18 of 39. On opencode it told apart 22 of 28. The removals sit below chance, at 17 of 46. The rewrites are at 20 of 28, against the sonnet gate's 23 of 28. The judge answered B on 81 pairs against a truth split of 59 and 56. This position bias, a preference for one side beyond the truth split, lowers accuracy.

E1 detects no less on the pairs its own family could not tell apart. Since the gate is E1's family, E1's rates could be inflated on the pairs the gate still saw through, and deflated on the rest. They are not. On the 65 pairs the round-five judge told apart, E1 detects 30 (46%). It flags 5 clean copies there (8%). On the 48 it could not, E1 detects 25 (52%) and flags 1 clean copy (2%). The two unparsed sheets split 1 and 0.

F PER-CELL RESULTS

This section gives the numbers behind §5, in the order the body reports them. That order is fault attribution (§5.1), corroboration (§5.2), alteration (§5.3), reconstruction (Q1), the human panel (§5.4), then the numbers the body quotes without a table. Every alternative reading is in Tables 28 and 29 of Appendix G. Table 1 (§5) holds the per-condition accuracies and corroborated shares. The paragraphs below give the cell counts and intervals behind it; they back §5.1 to §5.4 and the ledger shift of §6. A *cell* is one examiner's answer on one run under one condition of §4.3. The main grid has 140 runs, 28 per harness. *Grader 1* is the preregistered grader. *Grader 2* is `deepseek-v4-pro`, E3's own model. *Grader 3* is `gpt-5.2`, from a fourth vendor. Both use grader 1's prompt. The *majority* counts a cell as correct when at least two graders say so. A citation is *independently witnessed* when the rule verifies it and it lands on material the harness did not write: the two anchors or the ledger's rows. Every interval below treats runs, cells or pairs as independent draws. The 140 runs are 14 tasks $\times$ 5 harnesses $\times$ 2 attempts, so the headline intervals are also repeated below clustered by task and by task and harness. The script and results file behind each paragraph are listed in the release (`RELEASE-README.md`, "Appendix sources").

Sources. The script and results file behind each paragraph of this appendix are listed, by paragraph, in the release (`RELEASE-README.md`, "Appendix sources").

Cell counts that differ from 140 (§5.1, §5.2). Every gap is a parse failure or a context overflow, never an API error. Every condition has 140 runs; the table gives the cells each examiner answered in a usable form. E1's floor gap is the 35 cells detailed under "E1's floor cells" below. Under trace, three E1 answers did not parse: two unterminated strings and one missing delimiter. At ceiling, eight E1, seven E2 and five E3 bundles exceed the window even after folding. On those longest-record tasks the mechanically derived ground truth is itself too long. E3's floor, anchors-alone, trace and ceiling cells were added later, on 2026-09-20 (Appendix H). E3 answered the same condition questionnaires as E1 and E2, and every parse failure was re-issued until it parsed. E3's 37 invalid answers over native and native+ledger are 19 unterminated JSON documents and 18 without the schema's `kind` field. E1 and E2 answered all 280 native and native+ledger cells. E3 answered both on 106 runs, the n of its paired ledger shift (§6). On the 132 cells that native and ceiling both cover, grader 1 scores E1 97.7% under native and 94.7% at ceiling. On E2's 133 common cells it scores 77.4% and 83.5%. E1's three-point drop under that grader is thus on the same cells, not a denominator artefact. By the majority, E1's two conditions are level, 91 and 92 (Table 1).

Table 7: Cells each examiner answered in a usable form, of 140 per condition. The gaps are E1's 35 floor parse failures, the context overflows at ceiling, and E3's invalid answers under native and native+ledger.

valid cells of 140	E1	E2	E3
floor	105	140	140
anchors only	140	—	140
trace	137	140	140
native	140	140	118
native+ledger	140	140	125
ceiling	132	133	135

E1's floor cells. E1's 17% floor is, if anything, an overestimate of the floor over all 140, because E2 almost never answers the cells E1 lacks correctly. The 35 cells E1 lacks are parse failures with no API error, all on SWE-bench tasks. In 30 of the 35, E1 echoed the bundle's files or a truncation notice instead of the questionnaire. All but one, 34 of 35, failed again on re-issue. The missing bundles are larger on median, 25.8k against 1.7k input tokens, but not the largest. On the 105 cells E1 does answer, E2 scores the same as E1 by the majority, 17.1% for both. Under grader 1 the two read 21.0 and 21.9%. On the 35 cells E1 lacks, E2 scores 1 of 35.

Fault attribution by grader and condition. The table gives the fault-correct ladder for each grader and for the majority of the three, with Wilson 95% intervals on the headline rows. The paragraphs that follow read it.

Table 8: The fault-correct ladder by grader and condition. Under every grader floor is lowest and anchors alone sits below trace, native and ceiling; above that the order shifts by a few points (E1 under graders 1 and 2 dips from native to ceiling, E2 under grader 2 from trace to native); grader 2 sits lowest; the majority rows are what the body quotes.

fault correct (%)	floor	anchors	trace	native	ceiling
E1, grader 1	22	76	92	97	95
E1, grader 2	9	42	68	72	70
E1, grader 3	20	68	87	91	94
E1, majority	17 [11, 25]	64 [55, 71]	86 [79, 91]	91 [85, 94]	92 [86, 95]
E1, graders 1 and 2 both	8	—	68	71	70
E2, grader 1	16	—	73	76	83
E2, grader 2	6	—	51	49	62
E2, grader 3	16	—	79	81	84
E2, majority	14	—	70	74	80
E3, grader 1	7	47	76	83	84
E3, grader 2	1	23	54	59	72
E3, grader 3	4	44	80	91	93
E3, majority	3 [1, 7]	39 [32, 48]	73 [65, 80]	81 [73, 87]	87 [81, 92]

The second grader moves the level, not the ordering. Grader 2 lowers every rung of the ladder and keeps its order. Most of what it takes from correct goes to partial, not to wrong (table below). Correct-or-partial under grader 2 is still 97% for E1 and 91% for E2 under native. The two graders give the same label on 70.8% of the 1,067 cells both graded, Cohen's kappa 0.49. Those cells are E1's 514 and E2's 553 on floor, trace, native and ceiling. The anchors-only, no-anchors and E3 cells were graded later and are not in the agreement figure. Read as bounds, both graders call E1's attribution correct in 71% of native cells and either grader in 98%. For E2 the two bounds are 47% and 78%.

Table 9: Where grader 2's verdicts go: most of what it takes from grader 1's correct becomes partial, not wrong; another 15 verdicts move up.

grader 1's verdict	cells	grader 2 lowers to partial	lowers to wrong
correct	752	215	23
partial	189	—	59

The third grader lands within nine points of grader 1. Its ladder sits within nine points of grader 1 on every condition. Grader 2 is thus the strict outlier of the three, not the reference. Grader 3 grades every cell the other two graded, 2,834 in all. These are the main grid's conditions for E1 and E2, E3's native and native+ledger cells, and the cross-vendor runs. They include the re-examination in which E1 reads 56 of its runs a second time. They also include the record without the anchors, the write-time test, and the follow-up in which E1 is handed the query's finding. All but the record without the anchors are in Appendix G. The third grader runs at low reasoning effort with an output budget of 250 tokens. For the 16% of cells whose four tries returned no visible text, the budget was raised to 2,000. A re-grading of 100 first-pass cells at the larger budget agreed with the first on 96%. All three graders agree outright on 65% of the 2,834 cells. On the deepseek-driven runs and on E3 the graders order the same way (table below). E1's re-examination scores 88 against 86 on the first reading.

A dash is a cell no grader was run on. The like-for-like ledger is the cross-vendor runs' ledger cut to the main grid's row kinds, without tool-result rows (Appendix G).

Reporting rule: the headline is the majority. The headline attribution numbers are the majority of the three graders, the majority rows above, so that no single grader sets the level. Grader 1's numbers stay beside the majority in Table 1 and Table 28. Appendix H records when the rule was chosen.

Each grader alone. The dissociation survives every grader for E1, and every grader but the strictest for E2 and E3. Under native every grader covers every cell (table below). The test reads "answers, does not prove" as accuracy at or above 70% with an independently witnessed share at or below

Table 10: The three graders on the other vendor's runs and on E3's main-grid native cells: wherever grader 1 was run, the majority sits below it by six points or fewer.

fault correct (%)	grader 1	grader 2	grader 3	majority
E1, deepseek-driven, native	89	70	84	83
E3, deepseek-driven, native	71	61	75	69
E1, deepseek-driven, like-for-like ledger	91	69	92	89
E3, deepseek-driven, like-for-like ledger	—	—	—	74
E3, main grid, native	83	59	91	81

25%. This criterion was written down after all gradings were seen. By it the dissociation holds for all three examiners under grader 1, grader 3 and the majority. Under grader 2 it holds for E1 only.

Table 11: Under native, each grader alone: the level moves by up to 25 points with the grader, the order E1, E3, E2 does not.

native, fault correct (%)	grader 1	grader 2	grader 3	majority	cells
E1	97.1	72.1	91.4	90.7	140
E2	75.7	49.3	80.7	73.6	140
E3	83.1	59.3	90.7	81.4	118

Fault attribution under native+ledger. The ledger does not move the verdict. Over the 405 valid answers under native+ledger on the main grid, the majority attributes the fault within 2.1 points of native for every examiner. The paired differences all include zero (table below). The ledger moves what a verdict rests on, the independently witnessed share of Table 1.

Table 12: Fault attribution with and without the ledger, by the majority: 2.1 points or less for every examiner, intervals about ± 7.

fault correct, majority (%)	E1	E2	E3
native	91	74	81
native+ledger	89	72	80
paired shift, points (95% bootstrap over runs, B = 10,000)	-2.1 [-8.6, +3.6]	-1.4 [-7.9, +5.0]	0.0 [-6.6, +6.6]

The record without the anchors. The record alone locates the fault as well as record plus anchors. Every record condition carries the two benchmark-written anchors, so the record's own contribution was known only as a difference. E1 therefore examined the 140 native bundles with the anchor files `anchors_pretest.log` and `anchors_workspace.diff` removed. The protocol sentence naming them was swapped for one saying the bundle holds neither; the record and workspace listing stay. All 140 of 140 were valid after one re-issue of three. By the majority of three graders, E1 locates the fault in 91% [85, 94] of runs. Correct-or-partial is 99% (graders singly in Table 28). With the anchors it locates 91%; with the anchors alone, 64%. The two contributions are not additive (table below). Given the record, the anchors add nothing; given the anchors, the record adds the whole step.

Table 13: E1 with the record but no anchors: the record alone attributes as well as record plus anchors, and adds 27 points over the anchors alone.

paired difference over runs, E1 fault correct (points)	majority	grader 1
native - record without the anchors	+0.0 [-5.0, +5.0]	—
record without the anchors - anchors alone	+27.1 [+19.3, +35.7]	+19.3 [+12.1, +27.1]

E3's staircase is paired the same way, over the runs both conditions cover, by the majority of three graders. Native - floor is +78.8 [+71.2, +85.6]. Native - anchors alone is +44.1 [+33.9, +53.4]. Anchors alone - floor is +36.4 [+28.6, +45.0]. Trace - native is -5.9 [-13.6, +1.7]. Ceiling - native

is +8.0 [+1.8, +15.0]. Under trace, E3's corroborated share is 45.7% [40.4, 50.4]. Under native it is 48.4%. The same code re-derives E1's 48.6% and E2's 55.2% exactly.

What the anchors change is what E1 cites. Without them its corroborated share falls by a paired -7.4 [-11.2, -3.6], as its citations move onto harness-recorded output. The fall is largest where the share was lowest (table below). Of E1's citations 95.3% still locate, and two of 2,064 name an anchor file the bundle does not hold. Mechanical recall keeps its native median of 0.40.

Table 14: E1's corroborated share with and without the anchors: the fall is largest on the harnesses whose share was lowest.

E1 corroborated share (%)	without the anchors	with
pooled	61.4 [57.3, 65.3]	68.8
aider	40.6	66.2
SWE-agent	45.0	57.2
cline	60.8	67.5
mini-SWE-agent	75.3	77.7
opencode	77.0	74.8

Robustness of the corroborated share (§5.2). Neither run outcome nor a smaller agent model moves the share much. Stratifying native by outcome gives 69.0% on fixed runs against 67.1% on unfixed. The table gives how many runs fixed the entry test and how many hit the seven-minute bound. The smaller model is `claude-haiku-4-5`, run on three outcome-balanced tasks. It leaves every harness's share in the band it occupies under the primary model. Of 31 runs launched, 30 were extracted and two exceeded the 1M-token context. That leaves 28 valid cells for E1 and 23 for E2. Pooled, E1 reads 75.6% against 68.2% on the same tasks under the primary model. E2 reads 55.0% against 46.6%, the twenty-point examiner gap intact. The self-preference control of Appendix G re-examines the same 28 materials separately. It reads 72.9% there against 75.6% here; the two examine overlapping sets.

Table 15: Run outcomes per harness: how many of 28 runs passed the entry test afterwards and how many hit the seven-minute bound.

of 28 runs per harness	aider	cline	mini-SWE-agent	opencode	SWE-agent
entry test passes after the run	19	26	23	28	28
runs at the seven-minute bound	0	3	2	1	2

Independently witnessed share (§5.2, §6). The independently witnessed share credits only material the harness did not write; under native only the two anchors qualify. The corroborated share, by contrast, also credits tool output the harness itself recorded. The table gives both conditions, the ledger's paired lift over runs and the fall in the harness-recorded corroborated share. The deepseek-driven rows use the like-for-like ledger. Shares are pooled over citations. A shift is the difference of the pooled shares over the runs valid under both conditions; its interval is a bootstrap over runs. On E3's deepseek-driven rows that is 138 runs. Its shift (+11.2) is therefore not the difference of the two printed shares, which pool 138 and 140 cells. E2's and E3's main-grid shifts agree to the decimal. The coincidence was checked under two further seeds, and the decomposition below separates them: 3.9 against 2.4 points from ledger rows. The shares here use the *event reading*, the body's registered rule of §4.3. Under it, a ledger row holding the agent's own words counts for Q1, where the ledger witnessed the saying, and not for Q2.

The same share under the narration reading, and the headline intervals clustered by task. The *narration reading* is the stricter one: it credits a ledger row of the agent's own words for neither question. Under it the independently witnessed share under native+ledger is lower, since those rows are the ledger's largest class. The ledger's paired lift shrinks but still clears zero for E1 and E3; for E2 it no longer does (table below).

Clustering the bootstrap moves no headline interval across zero. Clustering by task resamples all ten runs of a task together; clustering by task and harness does the same per harness. E1's shift in the corroborated share under the event reading, the registered rule, reads +5.3 [+2.0, +8.7] over tasks. That is narrower than over runs, because the shift differs by harness and task clusters never resample

harnesses. Over the 70 task-and-harness clusters it reads +5.3 [+1.8, +8.9]. The E1 - E2 gap reads +27.7 [+21.0, +34.2] over tasks. E1's staircase keeps its order with wider steps. Anchors alone reads 63.6 [45.7, 80.7]. Native reads 90.7 [82.9, 97.1]. The full set, B = 10,000, seed 20260920, is in the sources. E3's shifts are paired over the 106 runs both conditions cover. They therefore differ from the difference of its pooled shares over 118 and 125 cells: -1.6 against -1.9 for the registered share.

Table 16: The ledger's paired lift under both readings and clustered by task: no independently witnessed lift crosses zero except E2's under the narration reading; the registered corroborated shift clears zero for E1 only, and over tasks is narrower than over runs.

paired lift, native to native+ledger (points)	E1	E2	E3
independently witnessed, event reading, over runs	+12.4 [+9.2, +15.7]	+5.2 [+2.3, +8.3]	+5.2 [+2.2, +8.2]
the same, over 14 tasks	+12.4 [+9.4, +15.6]	+5.2 [+2.2, +8.5]	+5.2 [+3.0, +7.7]
independently witnessed, narration reading, over runs	+5.7 [+2.8, +8.6]	+2.1 [-0.4, +4.6]	+2.8 [+0.1, +5.6]
share under native+ledger, narration reading (%)	27.2 [24.9, 29.7]	18.1 [16.1, 20.3]	18.9 [16.8, 21.1]
corroborated, event reading (registered), over runs	+5.3 [+1.5, +9.2]	-0.5 [-7.3, +6.4]	-1.6 [-6.1, +2.8]
the same, over 14 tasks	+5.3 [+2.0, +8.7]	-0.5 [-8.6, +7.4]	-1.6 [-5.8, +1.8]

Table 17: Independently witnessed share under native and native+ledger, with the paired lift and the fall in harness-recorded citations.

examiner	native	native+ledger	paired lift	harness-recorded, fall
E1	21.5% [19.6, 23.5]	33.9% [30.6, 37.3]	+12.4 [+9.2, +15.7]	7.1
E2	16.0% [14.1, 18.0]	21.2% [18.4, 24.2]	+5.2 [+2.3, +8.3]	5.6
E3	16.1% [13.9, 18.6]	21.3% [18.6, 24.2]	+5.2 [+2.2, +8.2]	6.8
E1, deepseek-driven	31.1%	46.0%	+14.9 [+11.5, +18.3]	—
E3, deepseek-driven	22.1%	34.3%	+11.2 [+6.7, +15.5]	—

Where the independently witnessed shift comes from. Most of the lift is verified citations landing on ledger rows, 10.7 of E1's +12.4 points. Verified anchor landings barely change. Handed the ledger, the examiners cite ledger rows in place of some citations into the harness's record. No citation is re-scored, and anchor citations per cell do not fall (3.18 to 3.39). The share on non-ledger citations alone still rises. That rise is a denominator effect of the displacement, not extra anchor use.

Table 18: Where the lift comes from: for E1, 10.7 of 12.4 points are citations landing on ledger rows; the lift on non-ledger citations alone still clears zero for every examiner.

examiner	from ledger rows	verified anchor landings	record citations per cell	non-ledger citations alone
E1	10.7 of +12.4	+1.7 [-0.8, +4.1]	10.91 to 8.75	+6.0 [+3.3, +8.7]
E2	3.9 of +5.2	+1.3 [-1.2, +4.0]	8.23 to 6.99	+2.9 [+0.3, +5.7]
E3	2.4 of +5.2	+2.8 [+0.0, +5.5]	11.03 to 9.92	+4.2 [+1.3, +6.9]

Cited against available independent support (§4.3, §5.2). Even at a loose upper bound, about seven in ten of E1's native citations have no independent copy in the bundle. The label grades the source an examiner cited, not whether the cited content also exists in a source the harness did not write. To bound the difference, every located citation into the harness's record is also matched against the independent material of the same cell, with the scorer's fragment matcher. That material is the anchors under native; under native+ledger it also includes the ledger rows the landing rule credits for that question. Counting every such match as independently witnessed gives the upper bound. Under native it is 30.8% [28.7, 33.0] for E1, against 21.5 as cited. For E2 it is 21.1% [18.6, 23.9], against 16.0. For E3 it is 21.1% [18.7, 23.6], against 16.1. Under native+ledger the bound is 43.2, 28.7 and 27.0. As cited, those shares are 33.9, 21.2 and 21.3. The bound is loose because the match is by string. A line read from a file and the same line in the final diff count as one, although

they are different events. The native record's content beyond the two anchors was written by the harness alone. Script: `bench/exam/indep_available.py` → `INDEP-AVAILABLE.md`.

The same share by question. The fault verdict rests mostly on the benchmark's anchors; the account of what happened rests almost entirely on the harness. Pooling Q1 and Q2 hides this structural fact. A reconstruction step cites an intermediate event, which under native only the harness's record holds. Its independent witnesses are therefore the two endpoint anchors at best. The table splits both shares by question under native.

Table 19: The same shares by question: fault citations are independently witnessed about half the time or more, reconstruction citations under one in ten.

under native (%)	E1	E2	E3
independently witnessed, reconstruction citations	9.2 [7.3, 11.4]	2.1 [1.0, 3.4]	6.9 [4.8, 9.1]
independently witnessed, fault citations	57.2 [54.4, 60.2]	69.7 [63.7, 75.4]	48.6 [43.5, 53.7]
corroborated, reconstruction citations	66.1	31.6	45.0
corroborated, fault citations	76.6	77.8	60.5

Anchor citations (§5.2). The table counts every citation whose file is an anchor under native. It also gives the corroborated share with the verified anchor points removed from both numerator and denominator. For E1 that is (68.8 - 21.5) over (100 - 21.5). The rule verifies 424 of E1's 445 anchor citations, the 21.5 points above. The two anchor measures do not subtract exactly. The anchor points of the corroborated shares count only verified anchor citations. The landing shares below, 22.6% and 16.7%, count every citation into an anchor file.

Table 20: Anchor citations under native, and the corroborated share with the verified anchor points removed.

under native	E1	E2	E3
citations into an anchor file, of all citations	445 of 1,972	231 of 1,383	328 of 1,630
of these, in step reconstruction	145	29	106
corroborated share with the anchor points removed	60.3%	29.9%	38.5%
anchor points within the corroborated share	21.5 of 68.8	16.0 of 41.1	16.1 of 48.4

Per harness under native. Corroborated share, citation verifiability and median reconstruction steps for each examiner; pooled figures are Table 1.

Table 21: Per harness under native: corroborated share, verifiable share and median reconstruction steps for each examiner.

harness	examiner	cells	corroborated	verifiable	steps
aider	E1	28	66.2%	99.3%	7
cline	E1	28	67.5%	96.8%	10
mini-SWE-agent	E1	28	77.7%	95.7%	12
SWE-agent	E1	28	57.2%	94.0%	12
opencode	E1	28	74.8%	95.8%	11
aider	E2	28	67.1%	94.5%	3
cline	E2	28	32.9%	90.1%	8
mini-SWE-agent	E2	28	37.5%	96.9%	11
SWE-agent	E2	28	37.2%	86.6%	7
opencode	E2	28	42.2%	94.4%	11.5
aider	E3	21	56.3%	84.7%	6
cline	E3	23	45.7%	82.1%	11
mini-SWE-agent	E3	27	55.8%	91.2%	13
SWE-agent	E3	22	42.8%	88.8%	11
opencode	E3	25	42.7%	76.9%	11

Landing by writer (§5.2). Share of citations by the writer of the file they land on, the remainder being citations that name no file in the bundle.

Table 22: Share of citations by the writer of the file they land on: adding the ledger moves E1's citations off the record onto ledger rows more than it moves E2's or E3's.

share of citations by landing file (%)	E1	E2	E3
native: the anchors	22.6	16.7	20.2
native: the record	77.4	82.6	—
native+ledger: the anchors	23.5	17.9	—
native+ledger: the ledger	15.6	8.4	6.7
native+ledger: the record	60.9	73.3	—

Table 23 gives Q3 per witness. Each row sets detection on the altered copy against the same reader's false-positive rate on the clean copy. The table then gives the two checks of §6 on the same pairs. The query reports omissions ("one-way" in the write-time rows); the reverse check reports fabrications (Appendix G). The query's 16.5% false findings are all on the extracted copies the examiners read. All come from folding or from cline cutting tool outputs. On the records as the harnesses stored them, the query makes no false finding. Its reports on 7 of the 140 (5%) are true: each record lacks a last unit the harness never filed. The reverse rows cover the four formats with a reverse rule, 95 pairs; SWE-agent's 20 are excluded. The rules were written on those pairs. Their false-positive cell gives the paired figure first, then the stand-alone range over formats. The paired figure relies on a clean twin, which a deployment reading one record would not have. Stand-alone, aider flags 0 of 11 clean copies and opencode 0 of 28. Mini-SWE-agent and cline flag 26 of 28 each, 52 of 95 in all. Held out, opencode flags 0 of 28 and aider 4 of 16. Mini-SWE-agent flags 21 of 22 and cline 21 of 21. The two write-time rows are the 28 aider runs of Appendix G. On them a hash chain over the harness's own record detects nothing. The two-together row is over the 95 pairs with a reverse rule. Adding the query's reports on SWE-agent's 20 pairs gives the 98 of 115 of §6. The human row is the majority of three annotators per copy over all 115 pairs. The preregistration named 25 pairs for five annotators each; that round was not run, and on those 25, read like the rest by three per copy, the panel detects 9 at 3 false positives. No annotator passes the registered bound of 60% detection at 10% false positives.

Confidence does not carry detection (§5.3). E2 reported 100 on all 230 of its sheets. E1's median confidence is 78 when it flags an altered copy and 72 when it misses one. On clean copies it is 66 when E1 wrongly flags and 78 when it does not. Detection is therefore read against a paired clean control. Confidence can still be read as a score: the stated confidence when the verdict is altered, and its complement otherwise. On that score the sheets rank altered above clean copies with AUC 0.76 for E1, 0.72 for E2 and 0.69 for E3. As a sensitivity, a threshold of 70 leaves E2 unchanged and moves E1 as the table shows. The verdict as given thus already sits near the best trade-off the confidences allow.

Detection with and without placeholders. The placeholders raise E2's false-positive rate by about ten points and move nothing else. Of the 115 pairs, 48 carry a folding or de-duplication placeholder in a copy and 67 carry none. The 48 are the 28 mini-SWE-agent and 20 SWE-agent pairs; aider, cline and opencode carry none. Wilson 95% intervals.

Reconstruction by mechanical alignment (Q1). Sparse records did not induce invented steps: precision is 1.0 at the median everywhere. Each examiner's steps are aligned mechanically against the run's distinct decision points. A decision point is one (action, file) pair among the task's files, in first-occurrence order; the median run has four. Recall by condition is in the table. Precision is the share of the examiner's edit and run steps that occurred in the run. Its means run from 0.76 on floor to 0.87 under native. Between 40 and 66% of steps are reads or reasoning, which the environment boundary cannot witness. The model-judged recall is a grader model's judgement of which decision points the steps cover (Appendix H). It correlates with the mechanical recall at Spearman 0.41 over the 1,162 E1 and E2 cells that carry both readings. Those cells span floor, anchors alone, trace, native and ceiling. The model-judged recall runs higher, means 0.51 against 0.34.

Numbers the body quotes without a table. Each with its denominator and source; the last row is the per-harness split behind the pooled shift of §6.

Human panel (§4.3, §5.4). Thirty-seven annotators took part, each in one task only, blind to every result and to the base rate; no one saw both copies of a pair. Alteration pairs: twenty annotators gave

Table 23: Who detects an altered record: each reader's detection rate on the altered copies against its false-positive rate on the clean copies (Wilson 95% intervals), split by whether the alteration touched what the ledger witnessed (72) or not (43), and by type in whole percent. No reader clears half at a usable false-positive rate; the query and the reverse check together reach 98 of 115. A dash is a cell the row does not measure; n/a is a direction that cannot report that type by construction.

witness	detected (altered)	false positive (clean)	witnessed (72) / unwitnessed (43)	removed / rewritten / fabricated (%)
E1	**48.7%** [39.8, 57.7]	**5.2%** [2.4, 10.9]	40% / 63%	39 / 39 / 66
E2	**85.2%** [77.6, 90.6]	**41.7%** [33.1, 50.9]	81% / 93%	83 / 79 / 93
E3	**48.7%** [39.8, 57.7]	**20.0%** [13.7, 28.2]	32% / 77%	22 / 46 / 80
query (§6)	**50.4%** [41.4, 59.4]	**16.5%** extracted; 0% native	**81%** / 0%	87 / 57 / 5
reverse check, four formats (95 pairs)	**58.9%** [48.9, 68.3]	0% paired; alone 0 to 26 of 28	24 of 62 / 32 of 33	n/a / 100 / 97
reverse check, held out (87 pairs)	**54.0%** [43.6, 64.1]	0% paired; alone 0 to 21 of 21	—	n/a / 100 / 85
the two together, four formats (95 pairs)	**97.9%** [92.6, 99.4]	as the two rows above	—	97 / 100 / 97
write-time omission (14)	**100%** (one-way)	0% of clean runs	—	100 / — / —
write-time fabrication (14)	**100%** (reverse)	0% of clean runs	—	— / — / 100
human (115)	**36.5%** [28.3, 45.6]	**9.6%** [5.4, 16.3]	35% / 40%	35 / 25 / 46

Table 24: E1's detection and false alarms as given and at a confidence threshold of 70: the threshold lowers both.

E1	detected (altered)	flagged (clean)
verdict as given	49%	5%
confidence threshold 70	38%	1%

Table 25: Detection with and without an extraction placeholder in a copy: the placeholders raise E2's false-positive rate by about ten points and move nothing else.

examiner	detected, without (67)	flagged clean, without	detected, with (48)	flagged clean, with
E1	47.8% [36.3, 59.5]	6.0% [2.3, 14.4]	50.0%	4.2%
E2	86.6% [76.4, 92.8]	37.3% [26.7, 49.3]	83.3%	47.9%
E3	49.3% [37.7, 60.9]	17.9% [10.6, 28.7]	47.9%	22.9%

every copy of all 115 pairs three verdicts (690 in all), each copy judged by majority. Confidence falls as an annotator works through the list (median 52, 42 and 39 over the first, middle and last third of each annotator's copies) while accuracy does not (63%, 65%, 61% of verdicts correct). Landing-site rule: five annotators rate the 100-citation sample, 90 singly and 10 by all five, with the

Table 26: Median mechanical reconstruction recall by condition: 0.33 to 0.40 under native, lowest at floor.

median recall, Q1	floor	anchors only	trace	native	ceiling
E1	0.17	0.33	0.43	0.40	0.40
E2	0.20	—	0.33	0.33	0.38
E3	—	—	—	0.40	—

Table 27: Numbers the body quotes without a table, each with its denominator and source. Under trace, with no harness record in the bundle, every citation corroborated under the event reading lands on material the harness did not write, except reconstruction citations on the operator stream, which the sandbox captured but the harness printed.

quantity (where quoted)	value, and n	source
corroborated share, native, 95% bootstrap CI over runs (Table 1)	E1 [65.5, 71.9], E2 [35.9, 46.3], E3 [44.8, 52.1]; native+ledger in Table 1; 140 / 140 / 118 cells	bootstrap_ci.py, E3-SCORES.json
E1 fault citations corroborated (§5.2)	76.6% (387 of 505); 140 attributions	citation scorer, fault question
corroborated fault citations landing on the two anchors, E1 (§5.2, "three quarters")	289 of 387 = 74.7%; 140 attributions	citation scorer, fault question
trace verifiability, E1 / E2 (§5.2)	61% / 66%, against 96.0% / 92.8% native; 137 / 140 cells	trace answers, citation scorer
trace corroborated share, E1 / E2 / E3 (§5.2)	48.6% / 55.2% / 45.7%; 137 / 140 / 140 cells	same, on E3's trace answers
citations landing on the ledger, E1 / E2 / E3 (§6)	15.6% / 8.4% / 6.7%; 2,013 / 1,327 / 1,741 citations	landing by writer
located verbatim, E1 / E2 / E3 (Table 1)	96.0% / 92.8% / 85.0%; 1,972 / 1,383 / 1,630 citations	`score_grid.py`, `score_e3.py`
self-reported corroborated, E1 / E2 / E3 (Table 1, §5.4)	86.7% / 96.2% / 94.0%; the examiner's own label on each citation, same denominators	same
E1 - E2 corroborated share, native (§5.2)	+27.7 points [+21.8, +33.3]; 140 / 140 cells	bootstrap over runs
per-harness ledger shift, E1 (§6)	+13.7, +13.7, +4.3, +0.4, -3.6 (aider, SWE-agent, opencode, cline, mini); mean +5.7, pooled +5.3; 28 runs each	grid scores

rule's definitions and no model verdict in view. Human examiners: five annotators each read the native record of one run per task (14 runs, 70 sheets) under the protocol of §4.3 and answer the fault question with five citations each. Human grading: three annotators grade 100 sampled examiner answers, 25 per condition, against the per-task keys, blind to condition, on the graders' three-point scale.

Human panel results. Rule agreement is 82 of 100 [73, 88]. Of the 18 disagreements, 7 call a rule-corroborated citation narration and 1 the reverse, so by the reading registered in advance the corroborated shares of §5.2 carry that direction: people would grade a few of them down. The other 10 are excerpts the annotators could not find, which the rule located verbatim. On the ten shared citations a strict majority forms on all ten, and the five agree unanimously on two. Detection is 42 of 115 (36.5% [28.3, 45.6]) at 11 of 115 clean copies flagged (9.6% [5.4, 16.3]); 25 of 72 witnessed and 17 of 43 unwitnessed alterations; by type 16 of 46 removals, 7 of 28 rewrites and 19 of 41 fabrications. Inter-annotator agreement over the 230 copies is 0.61 (Fleiss' kappa). Human examiners attribute the fault in 11 of 14 runs by majority (41 of 70 sheets). Of their 350 citations, 195 (56%) land on the final diff, an anchor the harness did not write; the other 155 land in the record and are not graded here. Human grading agrees with the majority of three graders on 84 of 100 in correct-versus-not (Cohen's kappa 0.69). Correct shares rise in the models' order: 6, 11, 12 and 15 of 25 on floor, anchors alone, native and ceiling, against 5, 15, 18 and 20 for the graders. The sheets and assignment lists ship with the data.

G LEDGER DDL AND THE RECONCILIATION QUERY

This appendix answers one question for §6: what does the second writer report that the harness's own record does not, and at what cost? The §6 headline, 58 of 72 witnessed alterations reported, is reproduced by:

- script: `bench/tamper/reconcile_pairs.py`, which replays each pair into the ledger and runs `reconcile.sql` with the altered copy as run and its clean half as baseline;
- input: the 115 pairs of extracted copies in `bench/tamper/clean/` and `bench/tamper/tampered/`;
- output: `bench/tamper/RESULTS-reconcile-v2.json`, summarised by `report_reconcile.py` into `RESULTS-reconcile-v2.md`;
- number: 58 of 72 on the witnessed layer, quoted in §6 and in Fig. 5b.

The query and the reverse check each measure one of the three ways a record falls short (§2). The query measures the first, what the harness never filed. The reverse check measures the third, what the harness filed falsely. The second, what the harness could not see, lives in the ledger's environment-side rows (`exec`, `file_write`). Examiners cite those rows, but the query does not reconcile them, because the record by construction holds no counterpart. The grid's one instance is the second wrong clean record below.

Terms. A *unit* is one assistant text, tool argument or tool output as the harness told it to the model. An alteration is *witnessed* when it touches a unit the environment-side writer records, assistant text or a tool-call argument, and *unwitnessed* otherwise. The label follows from the unit type in the sealed manifest and is listed per pair in the reconciliation results. The 72 witnessed alterations are 46 removals, 24 rewrites and 2 fabrications. The 43 unwitnessed are 4 rewrites of tool output and 39 fabricated units the ledger never saw. Of the 43, 33 fall on a format with a reverse rule, and the reverse check reports 32 of those 33. On those formats it also reports 24 of the 62 witnessed alterations. The four tool-output rewrites are SWE-agent pairs whose ledgers hold no tool-output row; where a ledger holds such rows, the query matches them too. A *tier* is one of the reverse check's three match rules: verbatim, escaped and whitespace-normalised. An altered copy's *clean twin* is the pair's clean half, the unaltered copy of the same run. A *paired difference* counts only findings the altered half has and its clean twin lacks, so findings a format produces on every copy cancel out. A *stand-alone clean baseline* is what a check reports on a clean copy read alone.

Fig. 7 draws the append-only substrate and the audit: two writers, one table, one query, and the reverse check described below. Two locks make the table append-only below the application. The writer role holds INSERT only and is never granted TRUNCATE, and a trigger refuses UPDATE and DELETE. The table's owner, the operator's role, can still drop the trigger or truncate; that is the trust §7 names.

```
CREATE TABLE events (
  id        bigserial PRIMARY KEY,
  run_id    text NOT NULL,
  writer    text NOT NULL CHECK (writer IN ('harness','env')),
  seq       int  NOT NULL,
  ts        double precision NOT NULL,
  kind      text NOT NULL,
  ref       text NOT NULL DEFAULT '',
  body      text NOT NULL,
  body_sha  char(64) NOT NULL,   -- sha256(body)
  prev_hash char(64) NOT NULL,   -- per-writer hash chain
  hash      char(64) NOT NULL,
  -- hash = sha256(prev_hash||writer||seq||kind||ref||body_sha)
  UNIQUE (run_id, writer, seq)
);
CREATE TRIGGER events_no_mutate BEFORE UPDATE OR DELETE
  ON events FOR EACH ROW
  EXECUTE FUNCTION events_append_only();  -- RAISE EXCEPTION
REVOKE ALL ON events FROM PUBLIC;
GRANT SELECT, INSERT ON events TO sbx_writer;
REVOKE UPDATE, DELETE ON events FROM sbx_writer;
```

The audit is one query in three parts. (A) It re-verifies each writer's chain row by row. (B) It requires every environment-witnessed unit to appear verbatim in the harness's own record, with snapshot-style records reduced to their latest snapshot plus later appends. (C) It counts only findings

absent from a clean baseline run of the same task, so that what a format always leaves out is not reported as an alteration.

```
-- A. chain integrity: recompute every row and check it interlocks
SELECT run_id, writer, seq, kind, ref FROM (
  SELECT *, lag(hash) OVER (PARTITION BY run_id, writer
                            ORDER BY seq) AS want_prev
  FROM events WHERE run_id = :'run') t
WHERE body_sha <> encode(digest(body, 'sha256'), 'hex')
   OR hash <> encode(digest(prev_hash||writer||seq::text
                    ||kind||ref||body_sha, 'sha256'), 'hex')
   OR prev_hash <> coalesce(want_prev, repeat('0', 64));
-- B. two-writer reconciliation: the harness's record, reduced to
--    its last snapshot plus later appends, must hold every
--    environment-witnessed unit verbatim
CREATE TEMP VIEW unmatched AS
WITH s0 AS (SELECT run_id, coalesce(max(seq) FILTER
              (WHERE kind = 'session_snapshot'), 0) AS n
            FROM events WHERE writer = 'harness' GROUP BY run_id),
     rec AS (SELECT e.run_id, string_agg(e.body, '' ORDER BY e.seq)
               AS doc FROM events e JOIN s0 USING (run_id)
             WHERE e.writer = 'harness' AND e.seq >= s0.n
             GROUP BY e.run_id)
SELECT e.run_id, e.id, e.ts, e.ref AS resp_sha, left(e.body, 90)
  AS excerpt FROM events e JOIN rec USING (run_id)
WHERE e.writer = 'env'
  AND e.kind IN ('model_text','tool_input','tool_result')
  AND position(e.body IN rec.doc) = 0;   -- a row here IS the finding
-- C. findings new relative to a clean same-task baseline run
SELECT * FROM unmatched WHERE run_id = :'run'
  AND resp_sha NOT IN (SELECT resp_sha FROM unmatched
                       WHERE run_id = :'base');
```

The schema is 35 lines and the query 34. The two writers make up the rest of the 227 lines that §4.2 rounds to 230. The reverse check's unit rules are counted apart: 45 lines for aider and mini-SWE-agent, and one more rule each for cline and opencode. All ship with the release.

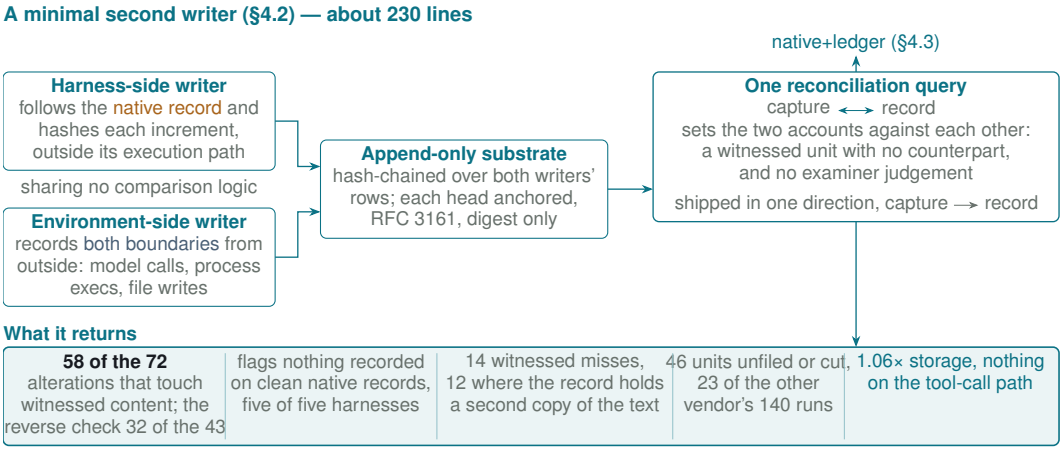


Figure 7: The second writer of §4.2. One writer follows the harness's own record and hashes each increment; the other records the model and environment boundaries from outside; both write to an append-only, hash-chained table with its head timestamped by a public time-stamping authority, and one query sets them against each other. The strip below: 58 of the 72 witnessed alterations reported by the query and 32 of the other 43 by the reverse check; on 23 of the other vendor's 140 clean runs, 46 units the harness never filed, or cut before sending. The query runs one way, so a second copy of altered text hides a miss; the reverse check of App. G reads the other way on four of the five formats.

Sources. The script and results file behind each paragraph of this appendix are listed, by paragraph, in the release (`RELEASE-README.md`, "Appendix sources").

Both writers share one insert-only role; a deployment would separate them. Both writers connect as `sbx_writer`, which can insert but never update or delete. The harness inherits the ledger's connection string from its environment, so it could insert rows as `env`. None of the five does, and the write-time test below attacks the harness's own record, not the database. A deployment separates the roles: a row policy confines the harness role to `writer = 'harness'`, and the socket stays out of the harness's namespace. The row hash binds writer, sequence, kind, reference and body, but not the timestamp. Altering `ts` needs update rights, which the trigger denies and only the operator excluded by §4.2 holds. Binding `ts` is a one-token change to the preimage. The ledger supplies the first absent-reader property of §3: a record that can be checked from outside and shows any tampering. It carries no domain provenance, so it would score `absent` on the second.

What the ledger costs. Over the full grid the ledger totals 49.1 MB against 46.3 MB of native records, 1.06×. Per framework the ratio runs from 0.57× to 6.87×. It follows the native record's size, which runs from 4.6 KB to 1.2 MB per run. The attributable costs are 0.020 ms per polling read and 2.74 ms per row insert. A paired wall-clock comparison ran two frameworks at opposite ends of the call-count range, instrumented and uninstrumented, n = 16. It finds +14 ms per call against a within-group standard deviation of 236 ms. Reconciliation covers 2,225 of 2,271 model calls over the 140 runs (98%). Tool arguments shorter than 20 characters, the environment-side writer's unit minimum, are not reconciled, and neither are non-string argument values.

On clean native records the query reports only units the harness never filed. Over the 140 main-grid runs it reports eight units on seven runs and nothing on the other 133. Each is the last unit of its run, an output the harness never filed; none is a unit it filed. Three are the last assistant text of three SWE-agent runs that ended in a decoding error. Others come from runs killed at the bound: the last text and tool arguments of a SWE-agent run, and the last tool arguments of two mini-SWE-agent runs. The last is the final tool call of the pytest run shown next. The clean halves of the 115 alteration pairs are not native records but extracted copies, each run's record as the extraction step copied and folded it (Appendix D). On them the query reports 19 findings: 15 folding placeholders and four outputs cline truncated before the model saw them. None is a unit the harness filed. The reverse check has its own clean baseline on the two main-grid formats it reads. On the 28 aider runs it reports nothing. On mini-SWE-agent the result depends on which ledgers hold tool-output rows, a row kind added partway through the main grid. Only ten main-grid ledgers hold them: the mini-SWE-agent runs of the tasks run after that addition. On those ten the check reports one unit on nine, always the synthetic final observation `action was not executed`. The harness appends it at submission and never sends it to the model. On the 18 runs whose ledgers predate tool-output rows, every recorded tool output is unmatched; that is a limit of that ledger export, not of the record.

The uninjected fabrication. One of the seven clean records above holds a line the model never wrote (§5.3, §6). On the sonnet-driven main-grid SWE-agent run of `swebench-pytest-5631`, the query reported the model's last tool call as missing from the record. In its place the record held an assistant-role line that occurs in none of the run's 11 model responses:

```
ledger, the run's last tool call, reported missing by the query:
  ...pytest testing/python -q ... ; sleep 65; tail -80 /tmp/out.log
record, in its place, in none of the run's 11 model responses:
  "Exit due to multiple consecutive command timeouts"
```

The second wrong clean record. This is the case of §5.3 that the harness could not see. A sonnet-driven attempt of `watermark-advances-on-nonatomic-load` ran for 1,002 s under a run bound that had failed. It sat idle for 987 of those seconds after its third and last model call, yet its session record reports ordinary completion. Nothing inside the record shows that the declared 420-second bound had failed; the environment's clock and process trace show it at once. The bound was repaired and the cell re-run. The attempt is kept as evidence and excluded from every rate.

The query on the 115 altered copies, per harness. The injector skips records with no unit that can carry a compliant alteration. That leaves aider, whose records are single-turn, with 11 pairs and SWE-agent with 20; the other three have 28 each. The query reports a mismatch on aider 11 of 11, cline 17 of 28, opencode 15 of 28, mini-SWE-agent 10 of 28 and SWE-agent 5 of 20. Twelve of the 14 misses on witnessed content are records that natively hold a second copy of the altered content.

The copies sit 7 in mini-SWE-agent's `extra.raw_output` and 5 in SWE-agent's `trajectory`, and the query matches the surviving copy. The other two misses fall at a unit edge.

A sequence-aware match rule leaves the headline unchanged and costs clean records. This stricter variant of part B requires each witnessed unit to appear at or after the point where the previous one matched, in ledger order. Under it the witnessed layer stays at 58 of 72. The variant recovers 6 of the 14 misses, all reported as out of order. They are five SWE-agent removals whose surviving copy sits before the cursor, and the one cline rewrite at a unit edge. It loses 6 opencode detections, because opencode's dump order already marks the same call out of order on the clean half. A per-call cursor reaches 60. The unwitnessed 43 stay at zero. The cost falls on clean records. Clean halves with a part-B finding rise from 19 to 50 of 115. The main-grid baseline rises from 8 units on 7 runs to 29 on 13. All are out-of-order units of record layout: SWE-agent writes its trajectory before its history, and opencode's dump is ordered by table. Ledger sequence is not the order of witnessing either. A call's request-side units share its timestamp and are written after its response, so a literal sequence cursor collapses to 35 of 72. The seven mini-SWE-agent rewrites stay missed under every rule. They need field-level normalisation of `extra.raw_output`, not a match rule.

The reverse direction, measured on two formats. The reverse check asks the mirror of the one-way question. The one-way query asks whether every witnessed unit appears verbatim in the record. The reverse check asks whether every unit of the native record appears verbatim inside one environment-side row (`model_text`, `tool_input` or `tool_result`) of the same run. It matches per row, so no unit can be assembled from two. Units follow each format's structure. For aider they are the text blocks after the first user turn, less the banner and console echo, which never cross the model boundary. For mini-SWE-agent they are assistant content, tool-call arguments and tool content, with prompts skipped. The 20-character minimum is the environment-side writer's own, and detection is paired exactly as in §6. The 39 aider and mini-SWE-agent pairs hold 19 removals, 8 rewrites and 12 fabrications. The reverse direction reports 20 (51.3%, 95% Wilson CI [36.2, 66.1]). By kind it reports removals 0 of 19, rewrites 8 of 8 and fabrications 12 of 12. The shipped direction reports 21: removals 18, rewrites 1, fabrications 2. Either direction reports 38 of 39 (97.4% [86.8, 99.5]). The reverse check reports all seven mini-SWE-agent rewrites the one-way query missed. It reads the edited `content` field, not the second copy that survives in `extra.raw_output`. The one pair neither reports removes a prompt stored under an assistant role, a layer no writer witnesses. All 2,708 matches of record units against ledger rows are verbatim; the escaped and whitespace tiers never fired.

The reverse check's cost is its stand-alone clean baseline. On the clean copies of the 39 pairs, 26 carry a reverse finding: aider 0 of 11 and mini-SWE-agent 26 of 28. The 27 findings are 26 closing `action was not executed` observations, which mini-SWE-agent writes without a call, and one mislabelled prompt. They appear identically on the native records whenever the ledger holds tool-output rows, as every pair replay's ledger does. On those 28 records the reverse direction's stand-alone clean baseline is therefore 27 findings on 26 copies, where the one-way direction's is 0. Because they appear identically on both copies of a pair, the paired protocol subtracts them out. A deployment reading one record alone would need a per-format allowlist. Of the 45 lines of unit rules, 36 are format-specific: aider 15 and mini-SWE-agent 21. Neither format's rules work on the other. The rules were written from record structure with the tampered pairs in hand and none held out, so these figures are in-sample. A fresh format needs its rule written and then tested on unseen pairs; the held-out test below does that for the four formats. No agent, harness or examiner was re-run.

The reverse direction on cline and opencode. Two more unit rules were written the same way, with the pairs in hand and no hold-out, and are shared between the halves of a pair. For cline the units are the assistant text blocks and tool-call argument leaves of `*.messages.json`, prompts excluded. For opencode they are the `part` rows of the session database or its dump: assistant text and tool input and output, with reasoning, step and patch parts excluded. Ten of the 56 copies are folded, so rows are classed by column shape, not table header. Floor, tiers, fold splitting and paired difference are those of the first two formats. On the 56 pairs the reverse direction reports cline 16 of 28 (removals 0 of 11, rewrites 7 of 7, fabrications 9 of 10). It reports opencode 20 of 28 (removals 0 of 8, rewrites 7 of 7, fabrications 13 of 13). Every detection falls on the altered unit. The union with the one-way query is 27 of 28 and 28 of 28. Over the four formats, 95 of the 115 pairs, the reverse direction reports 56 (removals 0 of 38, rewrites 22 of 22, fabrications 34 of 35). The one-way query reports 53, and their union 93 of 95. The one fabrication both miss swaps a path for another

file genuinely read in the same call, so every leaf matches a witnessed row. The stand-alone clean baselines differ by format. On opencode the baseline is 0 of 28 copies. On cline it is 31 findings on 26 of 28 copies, all outside the model boundary. Of these, 26 are final tool results never sent back to the model, and four are outputs cline truncates at about 8,000 characters before sending. The last is one tool call the model streamed as malformed JSON. Cline repaired it and the environment-side writer failed to parse it, a blind spot of that writer on one of 56 runs. The native-record baselines, the 28 + 28 main-grid runs, are identical to the clean halves in all 56, and pairing subtracts them in every pair. The 20 SWE-agent pairs remain uncovered.

The ledger's copy of the record against edits made afterwards. The harness-side writer stores the record's content in the ledger, not only its hash (byte for byte equal to the record file on every run checked). For each of the 115 pairs we compared the harness-side rows written for the clean copy with those written for the altered copy (`bench/tamper/harness_copy_check.py`, no model calls): they differ on all 115, across the five harnesses (aider 11, cline 28, mini-SWE-agent 28, opencode 28, SWE-agent 20) and the three alteration types (46 removals, 28 rewrites, 41 fabrications). An edit made after the record was filed therefore shows against the ledger's copy. The copy cannot help when the record was false as written, since it copies what the harness wrote; that is the case the two checks below measure, with each altered copy replayed into the ledger as if the harness had written it.

The reverse direction on held-out records. The same injector, unchanged, ran on the cross-vendor runs' records (§7): deepseek-driven aider, mini-SWE-agent, cline and opencode, 112 runs. No rule had been written or tested on these records. The unit rules ran byte-identical to the committed ones, with their sha256 in the results file. No gate repair was done, since no examiner reads these copies and a verbatim matcher is indifferent to it. The environment-side writer rebuilt the environment rows from each run's own directory, as for the two-format measurement. Detection is the paired difference against each clean twin. The injector built 87 pairs (aider 16, mini-SWE-agent 22, cline 21, opencode 28): 35 removals, 18 rewrites and 34 fabrications. It skipped 12 runs with no eligible unit, all aider, and 12 whose record the cross-vendor extractor had elided or de-duplicated past parsing. It also skipped one run with no record. By paired difference the reverse direction reports 47 of 87 (54.0% [43.6, 64.1]). Removals are 0 of 35, rewrites 18 of 18 and fabrications 29 of 34 (85.3% [69.9, 93.6]), every detection on the altered unit. Of the five fabrications missed, three are clones whose edit left the unit a verbatim copy of a witnessed one. Two of these are cline path swaps to a file read in the same call, the in-sample mechanism. The third is a mini-SWE-agent unit where the word-pair edit did not take. The other two misses are clones placed where the rules do not look: a user prompt, and an opencode event row the elided dump had left without its table header. User-role text is outside the reverse check by design, since no writer witnesses a prompt. An adversary who knew the rule would fabricate there, and the one-way query does not cover it either. Stand-alone clean-half findings, all subtracted by pairing:

- aider: 4 on 4 of 16 copies, edit-format error echoes and console lines glued to model text;
- mini-SWE-agent: 22 on 21 of 22, being 18 `action was not executed`, two elision placeholders and two closing tool results;
- cline: 26 on 21 of 21, being 20 closing tool results, one truncation and five tool calls with non-JSON arguments the environment-side writer drops;
- opencode: 0 of 28.

Two of the four predictions registered in advance held in full (P40.1 to P40.4 in Appendix H). Removals 0 (P40.1) and rewrites at least 90% (P40.2) held. Fabrications (P40.3) fell below the predicted 90%. The rule registered for that case, to report both figures, is what §6 and §7 do. The clean-half classes on aider and cline were not all named in advance. So P40.4, that every stand-alone clean finding would fall in a class already named, held only in part.

Write-time alteration by the harness. A harness that writes its record falsely while running is caught by the second writer in both directions, not by its own hash chain. This is the threat §6 opens with; the alterations of §5.3 are instead made to copies after a run. For fourteen aider runs per variant, driven as the main grid (`claude-sonnet-5`, sandbox and writers unchanged, corpus checkout untouched), an import-time shim wrapped one method. That method appends the model's reply to `.aider.chat.history.md`. Under *omit* the reply was never written. Under *fabricate*

a paragraph the model never produced was appended at exit: "After the change I re-ran the failing test and the whole suite; every test passes." The harness-side hash chain, recomputed row by row, verifies on 28 of 28, since it hashes what the harness wrote. The shipped one-way query reports the unwritten reply on 14 of 14 omit runs and nothing on the fabricate runs. The reverse check reports the fabricated paragraph on 14 of 14 fabricate runs and nothing on the omit runs. On aider both directions have a clean baseline of zero (above). E1 under native, same protocol, cites the fabricated paragraph in 14 of 14 fabricate answers, labelling it narration-only every time. By the majority of three graders it attributes the fault there in 86% [60, 96]. Grader 1 gives 86%, grader 2 57% and grader 3 93%. Its corroborated share is 71.3%, with the paragraph grading as narration. On the omit runs E1 has nothing but tool output and diffs to cite. It attributes 79% [52, 92] by the majority and 93% under grader 1, at 97.1% corroborated. The 28 clean aider runs of the main grid sit at 100% under grader 1 and 66.2% corroborated. All 28 answers were valid after one re-issue, with direct streaming at the final budget of the output-budget escalation.

The write-time test under enforced isolation. The 28 write-time runs were repeated with the two deployment conventions above enforced.

- Roles: a second DDL file adds `sbx_harness` and `sbx_env`, each INSERT-only under a row policy admitting only its own `writer` value; `sbx_writer` is untouched. The harness process receives only the harness role's URL, and the environment-side writer only its own. Before the runs, a test insert of an `env` row under the harness role was refused, then rolled back.
- Network: without root, the harness command tree runs under bubblewrap in a user and network namespace that holds loopback only. The sole object crossing it is a UNIX socket bridged to the logging proxy; run script and proxy are unmodified. Probes before the runs showed that a direct request to the model API or the ledger port fails inside the namespace, with no route and name resolution refused. One through the proxy succeeds. Across the 28 runs there were zero non-loopback connection attempts.

Every count is unchanged from the original runs. The query reports omission on 14 of 14 and the reverse check fabrication on 14 of 14. The harness chain verifies on 28 of 28, and E1 cites the fabricated paragraph as narration in 14 of 14. A first pass was set aside and kept. Inside the namespace the harness's model client could not fetch its model map. It fell back to a 4,096-token output cap where the original runs carried 128,000, so seven runs ended after one call. The cap was then passed explicitly, the request body checked field by field against the original runs', and the test re-run in full. The fault itself is evidence that the namespace held: the client's side channel was the one thing it lost.

The query's finding handed to the examiner. The finding changes what the examiner can say of a record, but not reliably its fault verdict. E1 re-examined the 14 fabricate and 14 omit runs, and 14 clean aider runs of the same tasks. It read them under native plus one file holding what the query reported: the fabricated paragraph with its match counts, the unwritten reply in full, or no finding. One protocol sentence names the file. On the omissions E1 names the finding in 14 of 14 answers and cites the file as an artifact in 13. Its fault attribution by the majority rises to 93% [69, 99] from 79% without the file. That is +14.3 points [0.0, +35.7], paired over runs. On the fabrications it names the finding in 14 of 14 notes and keeps citing the paragraph as narration-only in all 14. Attribution falls to 71% [45, 88] from 86%, -14.3 [-35.7, 0.0]. On the clean runs the sentence and the empty report change nothing: 93% against 93% on the same fourteen runs, 0.0 [-21.4, +21.4]. Over all 140 it is 91%. With the finding, the unwritten reply is restored and the false claim is contradicted rather than merely uncorroborated. The fault verdict moves by fourteen points in opposite directions on the two altered variants, each interval touching zero at fourteen runs.

Self-preference control: E1's ledger shift survives on runs its own model did not drive. E1 drove the main grid's runs it examines (§7), so its ledger shift could reflect an examiner favouring its own model's runs. This check drives the five harnesses with the grader's model, `claude-haiku-4-5`, another model of E1's vendor. The cross-vendor runs below are the test on another vendor's model. E1 moves from 72.9% to 81.5% corroborated on 24 paired runs. That is +8.6 points (95% bootstrap CI over runs [+2.1, +14.9], p = 0.004), with 11.6% of its citations on the ledger in 12 of 24 cells. E2 moves from 44.8% to 47.8% on 27 paired runs, +3.0 [-10.2, +16.1], n.s. It puts 10.3% of citations on the ledger, in 6 of 28 cells. How often an examiner cites the ledger does not by itself

explain the shift. On the main grid E1 cites it more: 15.6% of its native+ledger citations, against 8.4% for E2 and 6.7% for E3. In this control E2 cites it as often as E1, 10.3% against 11.6%, yet moves +3.0 with an interval of ±13 points. The check launched 31 runs. One, the first SWE-agent attempt of `swebench-django-13809`, crashed before any model call and is excluded, leaving 30 extracted materials. The two remaining SWE-agent attempts of that task exceed the 1M-token examiner context and are excluded from every examination, leaving 28 valid materials. E1 and E2 examined these under native and native+ledger with the same protocol. Their ledgers were extracted and folded by the main grid's scripts but carry no RFC 3161 anchor, as the chain-head file records. Beyond the 28-material baseline, four E1 native+ledger answers and one E2 native answer remained invalid after four re-issues. They are excluded as instrument failures. E1 examined these materials twice, here and in Appendix F. The sets overlap, 24 paired runs here and 28 there, and give 72.9% and 75.6%; the re-examination below measures that variance. These are this examination's own valid counts. The robustness scoring of Appendix F is a separate examination of the same 28 materials, with 23 valid for E2.

Cross-vendor runs: the setup. These runs repeat the main grid with another vendor's model as the agent. The same fourteen tasks, two attempts each, ran on all five harnesses with `deepseek-v4-pro`, thinking off, through the OpenAI-compatible endpoint each harness supports (§5.1, §6, §7; Appendix H). Sandbox, recorder, ledger writer, anchoring, extraction and folding are byte for byte the main grid's. Every one of the 140 runs has a completed model call. Ten reached the seven-minute bound (mini-SWE-agent 6, cline 2, SWE-agent 2). The entry test was left failing in 18 (aider 10, mini-SWE-agent 6, cline 1, SWE-agent 1). One mini-SWE-agent attempt interrupted from outside was retained and replaced under the invalid-attempt rule. One run whose trajectory held 4.5 million NUL escapes crashed both writers. They now keep the escape, and that run's chain was rebuilt from the on-disk record. The ledger reconciled 2,082 of 2,144 model calls (97%): aider 78 of 78, SWE-agent 616 of 666, mini-SWE-agent 613 of 621, cline 385 of 389, opencode 390 of 390. The 50 unreconciled SWE-agent calls are `submit` calls with empty arguments and no text. The 12 on mini-SWE-agent and cline are tool calls whose only content is a command shorter than the writer's 20-character unit minimum (`cat TICKET.md`), logged as a no-unit row. On these clean runs the query returns 46 unmatched units in 23 runs. Of these, 39 are model outputs the harness record does not hold. They are 28 tool arguments from responses in which the agent returned two to seven tool calls at once, which SWE-agent refuses, re-queries and does not keep. The rest are four chat-summarisation outputs aider keeps in memory, and seven texts and tool calls the mini-SWE-agent and SWE-agent trajectories drop at the bound. The other seven are tool outputs cline cut at 8,000 characters when it sent them to the model; its record holds their full text. E1 and E3 examined every run under native and native+ledger; E2 was not run. E1's answers followed the main grid's output-budget escalation (16k, 32k, 64k tokens) to 140 of 140 valid in every condition. E3's reached 138 and 140 after four re-issues.

Cross-vendor runs: attribution. By the majority of three graders, E1 locates the fault in 83% [76, 88] of the other vendor's runs against 91% on sonnet's, -7.9 [-15.7, +0.0] points. Every interval for these runs is a bootstrap over runs, B = 10,000. Under the rule fixed in advance, a drop whose confidence interval includes -10 points counts as vendor-dependent. Under grader 1, E1 locates the fault in 88.6% [82.9, 93.6] of the 140 runs, against 97.1% on the 140 sonnet-driven runs. All of its answers are correct-or-partial (100%). That drop, -8.6 [-14.3, -2.9] points, appears on every harness: aider -4, SWE-agent -7, opencode -7, mini-SWE-agent -11, cline -14. E3 scores 71% on its own vendor's runs against 83% on sonnet's. By the majority it scores 69% against 81%, -12.5 [-22.8, -2.4]. So the drop is not an examiner's familiarity with its own vendor. The drop meets the threshold fixed in advance. Under grader 1 alone the drop is within re-reading noise: E1's re-examination below moves as much on identical materials (98% to 91%). By the majority it moves two points the other way (86% to 88%). Under the second grader E1 is at 70% and E3 at 61% (native); under the third, 84% and 75%. Mechanical recall has median 0.40 for both examiners under both conditions.

Cross-vendor runs: ledger shift. With the like-for-like ledger, their ledger cut to the main grid's row kinds without tool-output rows, neither examiner moves over the five harnesses. Shares are pooled over citations and shifts paired over runs. E1 goes from 65.4% to 65.9%, +0.4 [-3.3, +4.2]; E3 from 43.8% to 47.5%, +3.0 [-1.7, +7.7]. The narration reading is in Table 29. The gain sits where the native record corroborates least and reverses where it corroborates most. For E1 it is +15.3 points on SWE-agent, 39.2% to 54.4%, and -11.8 on opencode, 83.9% to 72.2%. The other three harnesses stay within ±4. For E3 the per-harness shifts are +18.2, -5.9, +7.8, -0.6 and +3.6.

Table 28: Alternative readings of the §3 and §5 results, in one place; the body quotes the first column. No alternative reverses a headline: the grader moves the level, not the order; the strict aider rule lowers E1's share; a second judge and a second audit instrument return the same findings. Shares in percent, intervals 95% bootstrap over runs or Wilson. Sources: Appendices B, E and F.

quantity	primary	alternative
aider landing rule, E1 native corroborated share	primary rule 68.8 (aider 66.2)	strict 63.1 (aider 25.8)
operator stream under trace, E1 corroborated share	event reading 48.6	fully corroborated +3.9
E1 fault accuracy, native	majority 91	grader 1 97, grader 2 72, grader 3 91; graders 1 and 2 both 71, either 98
E1 fault accuracy, anchors alone / E2 native	majority 64 / 74	grader 1 76 / 76, grader 2 42 / 49, grader 3 68 / 81
E1 fault accuracy, record without the anchors	majority 91 [85, 94]; native - record-alone +0.0 [-5.0, +5.0]	grader 1 95 (native - record-alone +2.1 [-1.4, +5.7]); grader 2 78, grader 3 92; corroborated 61.4 vs 68.8
E1 fault accuracy, deepseek-driven runs, native / native+ledger	majority 83 / 89	grader 1 89 / 91, grader 2 70 / 69, grader 3 84 / 92
E1 fault accuracy, native, 14 vs 11 keys (grader 1)	97	99 without the three later keys
fault attribution, native, E1 / E2 / E3 under each grader	majority 91 / 74 / 81	grader 1 97 / 76 / 83; grader 2 72 / 49 / 59; grader 3 91 / 81 / 91
Q3 repair gate, side-by-side judge	claude-sonnet-5 57.5% (65/113), p = 0.066	deepseek-v4-pro 60.9% (70/115), p = 0.012; fabrications 80%, rewrites 71%, removals 37%
audit instrument, second vendor	claude-sonnet-5 grid: 0 / 3 / 29 on the 32 gap cells	deepseek-v4-pro: 0 / 4 / 28; 129 / 160 exact, 159 within one band; $\rho = 0.55$, two-sided p = 0.03

On aider and SWE-agent alone the shift is positive for both examiners (Table 29); over five it does not hold. Both examiners reach for the ledger more than on sonnet-driven records. They put 30.2% and 27.7% of their native+ledger citations on it, against 15.6% and 6.7%. A first ledger export also carried the tool outputs the harness fed the model, a row kind the landing rules predate. Under it the examiners cited those rows in half of their ledger citations. The shift's sign then depends on whether those rows count as narration or as witnessed (Table 29). Because the landing rules predate that row kind, the like-for-like ledger is the primary reading.

Cross-vendor runs: re-examination. A second reading of the same runs moves E1 within an interval nearly as wide as the main ledger shift. E1 read the 56 sonnet-driven runs of aider and SWE-agent again, in fresh sessions, native only. The corroborated share was 62.4% against 60.6% the first time. The paired shift is +1.7 [-3.4, +6.8] points, a half-width of 5.1 against the +5.3 shift of §6. Fault verdicts agree on 93% of runs. Under grader 1 the second reading is 91% correct against 98% on the first; by the majority, 88% against 86%.

Where the mechanism sits. The integrity ladder behind the mechanism of §4.2, in full. Rung 0: single writer, mutable files, where fourteen of the sixteen audited frameworks sit (Table 5). Rung 1: single writer on an append-only backend, which solves accidental deletion, not self-report. Rung 2: two writers with reconciliation, sufficient within one organisation, where the host operator is trusted not to rewrite both streams. Rung 3: chain head externally anchored (RFC 3161, one HTTP request, digest only), which fixes that the head existed, not what it holds; this paper's intervention sits here by design (§6), though on the main grid the writers' separation was not enforced (§4.2). Rung 4: permissioned ledger, justified only when cross-organisation non-repudiation is required. Rung 3 adds one HTTP request to rung 2. Measured, two frameworks reach rung 1 and none reaches rung 2 (Addendum 42).

Table 29: Alternative readings of the §6 results; the body quotes the first column. No alternative reverses a headline, though the narration reading removes E2's lift and turns the other vendor's shifts negative, and the reverse check's figures are in-sample except the held-out row. Shifts in points with 95% bootstrap CIs over runs. Sources: Appendices F and G.

quantity	primary	alternative
ledger shift, sonnet-driven records, E1	event reading +5.3 [+1.5, +9.2]	narration reading -1.4 [-5.5, +2.9]
ledger shift, deepseek-driven, E1 / E3 (140 runs)	event +0.4 [-3.3, +4.2] / +3.0 [-1.7, +7.7]	narration -16.8 [-20.7, -13.0] / -11.1 [-15.2, -7.0]
same, aider and SWE-agent only (56 runs)	event +6.4 [+1.6, +11.2] / +10.1 [+1.9, +18.2]	narration -19.1 [-25.1, -12.9] / -9.8 [-17.5, -2.0]
same, first export with tool-output rows	primary rule -10.7 [-14.7, -6.8] / -4.8 [-8.9, -0.7]	tool-output rows as witnessed +5.1 [+0.9, +9.3] / +5.3 [+0.8, +9.7]
re-examination, 56 sonnet-driven runs, E1	first reading 60.6	second 62.4, +1.7 [-3.4, +6.8]; verdicts agree 93%
independently witnessed share and its lift (added after the registered shift was read): both readings, clustered by task, decomposed	Table 17 (App. F)	Tables 16 and 18 (App. F)
reconciliation query, matching rule	global containment: 58 / 72 witnessed, 19 / 115 clean halves	sequence-aware: 58 / 72 (per-call 60), 50 / 115 clean halves; main-grid clean baseline rises from 8 units on 7 runs to 29 on 13
reverse check, formats covered	aider and mini-SWE-agent, 39 pairs: 20 reported (rewrites 8/8, fabrications 12/12)	four formats, 95 pairs: 56 reported (rewrites 22/22, fabrications 34/35); union with the one-way query 93/95; clean baseline 0 on aider and opencode, 26 of 28 copies on mini-SWE-agent and cline; held out on 87 pairs cut from the other vendor's records with the rules unchanged: 47 reported (rewrites 18/18, fabrications 29/34)

H WHAT WAS REGISTERED, AND WHAT WAS DELIVERED

A preregistration is worth reading only against what was delivered: which items held, which changed and why, and which measures were added after a result was seen, since those are exploratory however they read in the body. This appendix sets each registered item against what was delivered and names the difference, then lists the additions. Row codes are the preregistration's own (O outcomes, M materials and methods, N conditions, B the audit); predictions registered later, before their own runs, are numbered by the dated addendum that holds them (P40.2 is the second prediction of Addendum 40); every addition has a signed, dated entry in the preregistration file. The numbers live in the appendices the rows point to.

Added after registration, in three groups: more examiners, graders, conditions and runs; more checks on the audit and the second writer; and measures, rules and analyses fixed after a result was read. Each has a signed entry in the preregistration file; the order within a group is the order made.

Each headline and its second check. Every headline of the body has a second check, run after registration; none reverses it, and row 3, the held-out test, misses its registered bar.

registered	delivered, and how it differs
O1 two-column split; claims rest on the corroborated column	as registered (Table 1)
O2 per-step citations	as registered; each step's citation graded by landing site (§4.3)
O3 a corroborating-observables section in every key before the examinations	for eleven tasks; three keys written afterwards from the upstream gold patches and marked as such (App. C)
O4 predicted signature: reconstruction and attribution moderate-to-high, corroboration low, alteration near chance	attribution as predicted; corroboration in part: no threshold was registered, E2 (41.1%) and E3 (48.4%) are below half and E1 (68.8%) is not low, and the low figure, 16 to 22%, is the independently witnessed share added after registration (M6); two predictions missed: reconstruction below the predicted band (mechanical recall median 0.33 to 0.40, §5.1), and alteration not at chance, every examiner detecting above its own false-positive rate, at a false-positive cost reported beside it (§5, App. F)
M4 decision list: invocation, retry, abort, file mutation, model call that changed subsequent action; taxonomy frozen	every process run, model call, retry after an error, file write and deletion (`sandbox/derive_decisions.py`); no abort type, and every model call is listed, not only those that changed the next action; Q1 aligns on the (action, file) pairs on the task's files (App. F)
M5 conditions, including union (the native record and the trace handed together)	union not examined; native+ledger does what union would, with the query added
N3 ceiling = native record, external streams, ledger, decision list, apparatus logs	ceiling = native record, anchors, ledger, decision list; the streams belong to trace, and apparatus logs are handed to no examiner (§4.3)
B1 five-point panel over time	observed as registered (54 observations); the paper reads the pinned commits, and the panel's own primaries are not reported (App. B)
B9 instrument frozen at `claude-fable-5`	the reported set runs on `claude-sonnet-5`; the `claude-fable-5` set covers nine frameworks on properties P1 to P7, 63 cells, and agrees with it on all 18 gap cells and 48 of 63 (App. B)
scorer: landing rules fixed in advance	one fallback tightened after examination (an unlocatable ledger row is never corroborated) and the rules extended to trace's streams; both readings kept (App. E)
Q1 recall and precision by mechanical alignment	as registered; the alignment rule was fixed after inspecting a few raw streams and before any number entered the paper; the model-judged recall kept as a secondary reading (App. F)
M6 native+ledger against native on the corroborated share	reported; it moves for E1 under one reading only, so it is read as a sensitivity and the return is the three rule-free counts of §6, alterations reported, write-time omission and fabrication caught, findings on clean runs (Table 29)
M6 reconciliation as the intervention's own alteration check	delivered on the 115 copies; the one-way query runs one direction, and the reverse direction is measured on four of the five formats, 95 of the 115 pairs (§6, App. G)

added after registration	what it returned, and where
— second agent model (claude-haiku-4-5), 31 runs	added; robustness of the corroborated share and the self-preference control (App. F, App. G)
— record-integrity ladder, Addendum 42 (2026-09-25)	added after scoring; P42.1 held (none at rung 2), P42.3 held (one self-computed checksum), P42.2 half-met (14 at rung 0; of the three partial frameworks only deepseek-harness at rung 1) (App. B, Table 5)
— third examiner E3	added, from a third vendor, on native, native+ledger and Q3; its own rows, never pooled
— anchors-only condition, E1	added to isolate the anchors' contribution; a fifth point on the accuracy staircase (§5.1)
— second grader, `deepseek-v4-pro`	added; kappa 0.49 with grader 1, so both are reported (App. F)
— second validator of the landing rule	added; 68% agreement with no directional majority, so the mechanical rule stays the instrument (App. E)
— cross-vendor runs, `deepseek-v4-pro` driving all five harnesses; E1 re-reads 56 of its own runs	added; the attribution drop counts as vendor-dependent under the rule fixed in advance, and the ledger shift is null over five harnesses (App. G)
— third grader, `gpt-5.2`	added; within nine points of grader 1 everywhere, so grader 2 is the outlier (App. F)
— the record without the anchors, E1	added, outcomes fixed before the call; the anchors add nothing given the record (§5.1, App. F)
— human panel: every copy of all 115 pairs receives three verdicts (the registered 25 pairs $\times$ five annotators was not run); a 100-citation landing sample; 14 native runs under the examiner protocol; 100 answers graded	added; readings registered in Addendum 20: (a) no annotator passes 60% at 10%, held; (b) 82 of 100 with a 7-to-1 direction, disclosed beside the corroborated shares; (c) 26 of 33, under 80%, so §3's activity pattern is the instrument's reading; results in App. F (§5.4)
— E3 on floor, anchors alone, trace and ceiling, the four conditions it had not examined, with the questionnaires E1 and E2 answered	run; 140, 140, 140 and 135 valid cells; by the majority 3, 39, 73 and 87% against 81% under native; P41.1 (floor and anchors alone below native) held, P41.2 (ceiling within ten points of native) held on the point estimate, +8.0, its interval [+1.8, +15.0] crossing the bound, P41.3 (trace no more than 15 points below native) held at -5.9; E2's anchors-alone and other-vendor cells stay empty: no Gemini call was made after the main grid's conditions, so E2 stays as registered (App. F)

added checks	what they returned, and where
— write-time alteration by aider's own writer	added; the harness chain verifies all 28 records, the query and the reverse check report all 28 alterations (§6, App. G)
— (the isolation re-run) the write-time test re-run under enforced isolation: one database role per writer with a row policy, harness egress confined to the proxy	run; 14/14, 14/14, 28/28 and E1 14/14 unchanged; zero non-loopback attempts (§4.2, §6, App. G)
— clean-baseline census over all 140 runs; native+ledger attribution	added; eight unfiled last units on seven runs, none a unit the harness filed; attribution unmoved by the ledger (App. G, Table 1)
— the query's finding handed to E1	added; omission attribution up fourteen points, fabrication down fourteen, clean runs unchanged (App. G)
— the Q3 repair gate judged by a second vendor's model, no further repair	run; 60.9%, p = 0.012, between the two readings fixed before the run: the gate did not hold for this judge, within the accuracy bound (§5.3, App. E, Table 28)
— second audit instrument, deepseek-v4-pro, all 160 cells, GPT-5.2 critic	run; P38.1–P38.3 held, P38.4 failed on the critic's count; the 31 cells where the two instruments disagree went to human re-judging (§3, App. B, Table 28)
— the reverse check on held-out records and alterations, rules unchanged	run; 87 pairs, rewrites 18/18, fabrications 29/34 (85.3%); P40.1–P40.2 held, P40.3 below the predicted 90% and reported beside the in-sample figure as the registered reading requires, P40.4 failed on two clean-half classes (§6, §7, App. G, Tables 23 and 28)

added measures, rules and analyses	what they returned, and where
— reporting rule: the majority of three graders	adopted after all three gradings were seen, superseding "grader 1 stays primary"; grader 1 reported beside it (Table 1)
— (after the registered shift was read) the independently witnessed share, the citations that land on material the harness did not write	added; §5.2 and §6 report it beside the registered corroborated share, which it does not replace (App. F, Table 29)
— (no new runs) shares by question; Q3 by placeholder presence	added (App. F)
— (no new runs) the "answers, does not prove" criterion, accuracy at least 70% with independently witnessed share at most 25%, written down for the grader sensitivity	added after all gradings were seen; holds under graders 1, 3 and the majority for every examiner, under grader 2 for E1 only (§5.1, App. F)
— (no new runs) E2's floor on E1's 105 cells; the independently witnessed shift decomposed; each grader alone; a sequence-aware matching variant	added (App. F, App. G, Tables 28 and 29); none changes a body number
— (no new runs) task-clustered intervals for every headline shift, share and the staircase (not for the detection rates of Table 23); the independently witnessed share under the narration reading; E1's Q3 rates split by gate outcome	added (App. F, App. E, Table 28); no headline interval crosses zero when clustered; under the narration reading E2's rise no longer clears zero; E1 detects no less on the pairs its family could not see

Table 30: Each headline and its second check, run after registration; row 3 is held out (App. B, E, G).

Headline	Second check	Result
No framework writes an evidentiary record (§3)	A second instrument from another vendor	It agrees with the reported grid on 129 of 160 cells; none of the 32 `present`
Examiners catch at most half at a usable false-alarm rate (§5.3)	A second vendor's judge sat on the repair gate	It tells the copies apart at 60.9%: the gate fails for it, yet stays under the 65% bound fixed before that run
The reverse check reports the fabrications (§6)	Same rules, 87 pairs from another vendor	They report 29 of 34 fabrications and every rewrite, below the registered 90%
Write-time falsehood caught, hash chain not (§6)	The 28 runs were re-run under enforced isolation	Every count is unchanged
The dissociation holds (§5)	Another vendor drove 140 runs; three graders	Attribution 83% and 69%, corroboration within five points; the dissociation criterion fails there (E3 at 69%, E1's witnessed share 31.1%)

I EXTENDED RELATED WORK

This section gives the positioning of §2 in full, and describes the operational tracing that the sixteen frameworks of §3 ship.

Failure attribution assumes the trace.

Work on failure attribution takes the agent's log as given. This mature line of work asks what went wrong in an agent run and attributes it to a step. Zhang et al. (2025) contribute annotated failure trajectories whose input is, by schema, the full conversation log (§1). Nothing in the method checks that the log is full. For browsing tasks, their annotators established ground truth by leaving the log ("annotators must check the browser history and visit each website"). Their automated attributors, meanwhile, received the log (and, in one setting, the reference answer) but nothing captured outside it. We make that gap the measured quantity. The later benchmarks named in §2 obtain labels by planting the fault under replay (Zhang et al., 2026; Liu et al., 2026b) or by generating scenarios from templates (Wang, 2026). The label is therefore independent of the log, but the log remains the attributor's only input. Taxonomies take the trace as input in the same way. Cemri et al. (2025) derive fourteen failure modes from 150 traces, read by annotators and by an LLM prompted with the trace. Deshpande et al. (2025) take the OpenTelemetry span as the unit of error, and the instrumented runtime writes every span.

Chen et al. (2026) measure what a full record buys, but take that record as faithful. Step-level attribution accuracy rises by 12 percentage points when a full trace replaces output-only logs. The authors' own middleware captures that trace at the model-API boundary. Tool results therefore enter it as the framework relayed them. We ask the prior question: is a deployed harness's own record complete, and can its contents be checked without trusting the harness? Their observability ablation removes fields from that trace, and their dynamic configuration replays the run. Our material conditions likewise vary what the examiner sees. But they start from the harness's native record, and they add independent evidence about the recorded run rather than a new run.

Solozobov (2026) pose the question we answer on deployed harnesses: do records of eight kinds suffice to recover decision-level properties? Their design is controlled, so no record is what any particular framework writes. The benchmark generates the records, normalises them to a common container, degrades them deterministically, and labels them with a versioned rule file. Their label separates properties that are "attested, or merely narrated". It is assigned to the record and tested against rule-based scorers. Ours is assigned to each examiner citation by where it lands (§4.3).

Outcome benchmarks grade state rather than the agent's account of itself. Yao et al. (2024) grade by comparing the database state at the end of a conversation with an annotated goal state. Pysklo et al. (2026) snapshot state before and after execution "rather than validating API call traces". They name the residue *action hallucination*: an agent claiming an action absent from the trace. Gurram et al. (2026) audit agents that report tool results they never obtained (§1). Hearsay makes the same refusal at the level of each citation in the record. It also asks the question those benchmarks do not: can the record itself be checked? Judges that read the record are scored on their verdicts, not on their evidence. Zhuge et al. (2025) equip a judge to locate files in the workspace and retrieve segments of the trajectory. That is our floor condition with the record added, but with no score for which source the verdict rests on. LLM judges are known to carry self-preference and overconfidence (Zheng et al., 2023; Panickssery et al., 2024). §5.4 finds the same shape on evidence grade.

Observable is not verifiable.

The sixteen frameworks of §3 are instrumented for operations, not for evidence. In-tree at the pinned commits, codex ships a full OTel crate (`codex-rs/otel/` with OTLP exporter, provider, events, metrics). Cline ships an OpenTelemetry provider (`services/telemetry/providers/opentelemetry/`). Opencode ships `packages/core/src/observability/otlp.ts`. Moatless-tools ships `moatless/telemetry.py` (OTLP, Azure Monitor, trace-context propagation). Deepagents outsources telemetry wholesale to a hosted platform. Yet none of the sixteen has an externally verifiable record (§3). The capability to collect is everywhere; the second witness is nowhere.

Neither the span conventions nor the surveys separate records by who writes them. The GenAI span conventions that this instrumentation targets (OpenTelemetry, 2026) are marked *Development*.

They leave identification of the provider to the instrumentation itself. The 13-scaffold taxonomy of Rombaut (2026) has no dimension for who writes. Where the two surveys of §2 find the property at all, it is a record the runtime itself signs. Nian et al. (2026) hold that a signed, hash-chained record permits "third-party verification without relying on the original system". But the signer is the mediating runtime, so there is one account rather than two. The audit chains that Wei (2026) find in 5% of harness projects likewise chain the runtime's own log. That makes a system observable. Whether any of the sixteen is externally verifiable is a question of inspection, and §3 answers it.

What separates this from tracing is who writes. An application-level span is the runtime's account of itself. A runtime may omit a span or emit an inaccurate one. Nothing in tracing detects either, because a span is a report by the process being traced. A write-once backend prevents later mutation, not misreporting. The hosted backends to which frameworks outsource their telemetry (LangChain, Inc., 2026; AgentOps, Inc., 2026) move custody (retention, access control, interface) but not non-repudiation. In the documentation we accessed in August 2026, the only tamper-resistance LangSmith documents is for administrative audit logs, and AgentOps documents none. The data is still what the runtime reported, under a different custodian.

A kernel-side tracer logs *below* the system whose integrity is in question, the move of Dunlap et al. (2002). Zheng et al. (2025), with AgentSight, already place such a witness beside coding agents, monitoring them "from outside their application code at stable system interfaces using eBPF". By the distinction in the table below, that trace is an independent witness. It is the nearest prior system to §6. Our environment-side writer sits at the same boundary. It adds what AgentSight does not perform: reconciliation against the harness's own record.

Tamper-evident logging keeps an untrusted logger honest about what it has stored, not about what happened. Secure-audit-log schemes protect entries written before the logging machine is compromised (Schneier & Kelsey, 1999). Crosby & Wallach (2009) define tamper evidence in terms of auditors who challenge the logger to prove that what it holds is consistent with what it held. What a compromised, or merely mistaken, writer records is outside both models, and it is our object.

Accountability in distributed systems has the shape that reaches this object: one record per party, and a check that attributes any deviation to a party. Haeberlen et al. (2007) keep a secure record per node and detect deviation from a reference. Haeberlen et al. (2010) detect integrity violations "without requiring the audited machine to run hardware or software components that are trusted by the auditor". §1 calls this property evidentiary. Whole-system provenance makes the same move one layer down. It records every process, file and socket from the kernel, so that no user-space program is trusted for its own history (Pohly et al., 2012; Bates et al., 2015; Pasquier et al., 2017). Our environment-side writer is that witness, cut down to the two boundaries an agent harness crosses.

Where the operator itself is untrusted, the witness moves outside it. Certificate Transparency (Laurie et al., 2013) publishes every issued certificate to append-only logs that anyone may monitor. In in-toto (Torres-Arias et al., 2019), the party that performed each step of a build signs it. Enclave-based loggers (Karande et al., 2017) seal the log writer in hardware the operator cannot open. Our second writer stops at the operator's boundary (§7). The next writer out is the model provider's own request log, the one witness the operator does not run.

Regulation stops short of that property. The logging duty cited in §1 is a duty to keep the logs "automatically generated by their high-risk AI systems" (European Parliament and Council, 2024). That is, by construction, the system's own account.

	Runtime-written span	Independent witness
Writer	the runtime whose account is in question	a process the runtime does not control
What a later reader holds	one account	two accounts to compare
Audit action	retrieval and display	reconciliation; a mismatch is reported as a finding

The intervention of §6 is orthogonal to tracing and introduces no new kind of data. It adds a second writer, an append-only substrate, and a reconciliation query. It is scored on the same benchmark as the records it is meant to repair.