Counterweights: Influencing LLMs

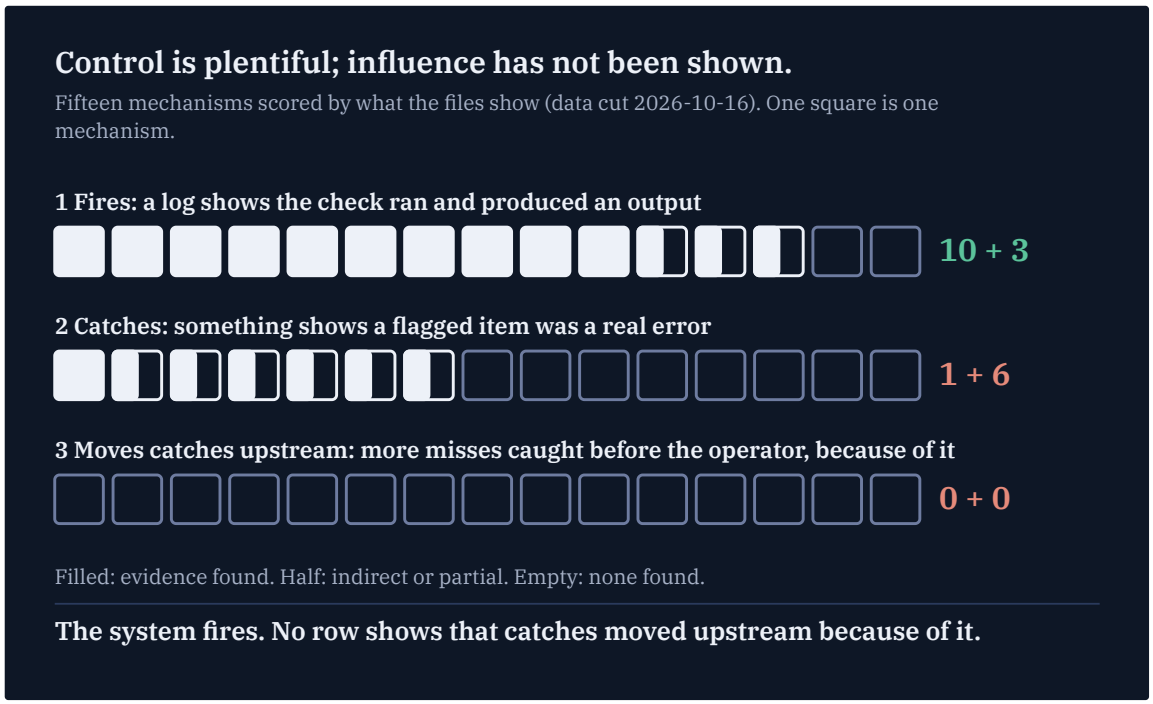
What can be controlled at the edges of a model that cannot be opened, and why that control cannot yet be shown to influence anything.

Berg Atkinson, Wolfberg LLC · berg@wolfberg.ai

Preprint. Not peer reviewed. Published 17 October 2026. Version 2.2, impersonal register and redrawn figures (supersedes v2.1).

Revised 2026-10-20: impersonal register, author block removed, figures redrawn. Internal identifiers moved to the appendix. Data, counts and claims are those of v2.1 (data cut 16 October 2026) and have not been refreshed.

Version 2.3, 30 October 2026: set in the house research-note format; the figures are shown on house plates and called exhibits. Text and numbers are otherwise unchanged from Version 2.2.



At a glance. Of the fifteen checks scored, ten show evidence of firing, which is control exercised, and three show it in part. One check, the human operator, has the cleanest record of catching real errors, and even that record leans on the agents: 219 of the 378 catches credited to the operator rest on the agent's word. None shows that catches moved upstream of the operator because of it. The share of misses caught before they reach the operator rose from 29% in June to 63% in October, but the go-lives overlap and there is no comparison arm (Section 4.7). No mechanism meets all three clauses of the specification, because none clearly meets (b). One meets (a) and (c) on the strict reading of (a), the operator, and four on the literal reading, three Stop guards and the operator. **Counts from the live and archive ledgers to 2026-10-16 11:54Z, the public register build of 15 October, the joint ledger and the project board read on 16 October (Section 4). The scoring is a judgment applied to a stated rubric, an argument and not a measurement (Section 4.2).**

IN PLAIN TERMS

An AI agent stops when it can defend its answer, not when the answer is right. Written rules did not fix that: the practice's own series counts seven misses of one act on a single day in July, recorded on the practice's boot page (background paper 3). Conjecture: that is one more way a model leans toward being accepted. This note states a thesis and tests it instead of assuming it. All you can do is influence an LLM. The practice described here, Wolfberg LLC, is one human operator and a team of AI agents. The models are rented and their weights are never touched, so the model is a box that cannot be opened. What can be controlled is the edges of the box: what leaves it (the Stop, a refusal, the one forced retry, a verifier on the agent's own claims) and, in part, what enters it (the context it loads, the rules, injected checks, the hooks that decide what it can reach; not the content of the tool results it fetches). A counterweight is a control point at one of those edges, external to the agent's own writing, firing on real errors rather than on a confident tone, and arriving at the moment the agent stops. What that control buys inside the box is influence, and how much of it can be shown is the question this note asks.

The design thesis, which the note tests, is that a check inside the loop that is answering can be talked past, and a separate process that writes to a ledger the producer cannot close is a gate (background paper 4 puts a related rule: an instrument the measured party operates will rot). A refused Stop is control over one turn: it holds the output back once, the retry it forces cannot be blocked, and whether that retry is better is influence. Input control is only partial: tool results (fetched pages, file reads) enter on the model's own choice, so what can be governed is what the model can reach, not the content it gets back. That is the gap the planned checks are meant to narrow (Section 3.2).

The system places, or plans, a control point where work would otherwise pass into something that counts: the Stop (the agent says it is done), Launch, Ship, Promote and Publish, plus the action, the merge, the canon write, checkpoints during work and nightly checks after the fact. By the classification used here (rule and reasons in Section 2.1 and Table A1), ten of the fifteen checks scored sit at the output edge, two at the input edge, two audit the harness or pipeline, and the operator works at both. Each sits in one of three places: inside the agent's own session (hooks that read the transcript and can send the agent back once), in a separate process (a reviewer that runs beside the research agent, readers on pull requests, a launch gate, nightly checks), or in a person (the operator reads the answer, merges every pull request and gives the go). What they do is written to ledgers that feed a public register at wolfberg.ai/register and a local database. The system was built up in clumps, from a kill check on 17 July to a launch gate and wiring checks on 14 to 16 October, and fourteen planned or proposed items (seventeen counting each separately: eleven unbuilt additions, two measurement fixes and one offline design) are still to come. Adding them one at a time, ideally switched on and off by session, makes each addition its own test.

Only then does the note ask how much influence that control has shown, at three levels of evidence. A check FIRES if a log shows it ran and produced an output (a refusal, a verdict, a list); that is control exercised. It catches if something shows a flagged item was a real error. It moves catches upstream if, after it went live, a larger share of misses was caught before they reach the operator, by the agent itself, a peer instance or an instrument; the last two rungs are the influence. A ruling of 16 October (a session statement, not logged to a file) set shipped errors aside as the wrong headline measure for this practice, and this is the measure that replaces it. Of the fifteen mechanisms read, ten show evidence of firing and three in part. The newest to be scored, the evidence guard, shows none: it has written no row, and its own channel entry says it writes none until the operator registers it. One check, the operator, has the cleanest record of catching real errors, and even that leans on the agents: 219 of the 378 catches credited to the operator rest on the agent's word. Six more show catching indirectly (a reviewer's finding that the agent which wrote the document then applied, for example). None shows that catches moved upstream because of it. The share caught before it reached the operator rose from 29% in June to 63% in

October, but the checks went live together, the field that says who caught a miss is written by the agent being measured, and there is no comparison arm, so it cannot say which check, if any, did it. The registered test that would settle this sits in the board's backlog.

Four hooks are wired at the Stop, the moment the agent says it is done. Three of them have fired there and so meet the moment clause; the fourth, the open-check guard, has run 1,704 times and never refused. One of the three, the claim guard, also runs a verifier with no model on each Stop's certificate: it refused 27 of the 356 certificates that listed a claim (7.6%) and turned the agent back 22 times; the other 1,046 of 1,402 listed no claim, so it passed them with nothing to open. All four run inside the session that wrote the answer, so whether they count as external depends on the word. As code outside the model they do. As independent of the producer they do not. Counted both ways, one mechanism is independent of the producer and at the Stop (the operator reading the answer, which does not scale and has its own drift), and four are external to the output stream and at the Stop (the three hooks and the operator). A second class of failure is invisible to every check that reads answers: output nobody reads, a component built and never connected. A board note of 16 October records that the easy stuff to miss is stuff buried in code, and two mechanical checks for it were ruled that day.

Position: a counterweight in the agent's own loop is a guardrail, and the evidence here is what a guardrail looks like from outside, many firings, a retry it cannot block, and no way yet to tell prevention from nuisance. In the thesis's own terms control has been shown and influence has not. The next step is the registered test, not another hook. The program's pre-specified null still stands: if recurrence of the tracked error classes does not move, the program closes.

WHAT TO COPY

1. Make the agent write a claim certificate when it stops, and check every pointer with code, no model.
2. Count a pass with zero claims as a non-check, not a pass.
3. Turn new checks on one at a time, or on and off by session, so each one is its own test.
4. Before merge, test that every new output has a reader.

GLOSSARY

- **Stop.** The moment the agent says it is done; the moment clause (c) of the specification names.
- **Commitment moment.** The point at which work would otherwise pass into something that counts: the Stop, Launch, Ship, Promote and Publish, plus the action, the merge, the canon write and checkpoints during work.
- **Latch.** A flag is honored after a refusal, so a Stop guard turns a turn back at most once; the retry it forces cannot be blocked.
- **Certificate.** The record the claim guard writes at a Stop, listing each claim the final text makes; one that lists no claim is a pass with nothing checked.
- **Verifier.** A verifier with no model: for each claim in a certificate it re-opens the transcript and the verify store and checks that the pointer exists and is what it says.
- **Refusal join.** The repository's own join of guard refusals to the operator's corrections, by agent and a 20-minute window, not by turn; it withholds every rate while the correction channel is dry.
- **Upstream share.** The share of misses caught before they reach the operator, by the agent itself, a peer instance or an instrument, against those the operator catches.
- **Register.** The public miss register at wolfberg.ai/register, rebuilt nightly from the session-health ledgers.

- **Stop-moment challenger.** The challenger that would act at the Stop, with its on/off arm machinery: built, imported by no live hook, with every agent UNASSIGNED.
- **The program's H3 and the plan's H3.** The program's: a reversal is not a correction. The plan's (plan v0.5): a refusal buys one turn-back, and the retry is verified but cannot be blocked.

ABSTRACT

An AI agent stops when its answer is defensible, not when it is correct, and written rules did not change that. The thesis tested here is that all you can do is influence an LLM: the models are rented and cannot be opened, so control reaches what leaves them and, in part, what enters them. The note describes the counterweight system that one practice (Wolfberg LLC, one operator and an AI team) built on that footing: control points at the commitment moments, inside the agent's session, in separate processes and in the operator, writing to ledgers and a public miss register, added in ten joint go-lives from 17 July to 16 October 2026. To 2026-10-16 11:54Z, three Stop guards refused 272 times, followed by 260 latched retries that by design cannot be blocked, and a verifier with no model refused 27 of the 356 stop certificates that listed a claim (7.6%). Of fifteen mechanisms, ten show evidence of firing and three in part; one, the operator, has the cleanest record of catching real errors and six show it indirectly; none shows catches moving upstream of the operator. The share of misses caught before they reach the operator rose from 29% in June to 63% in October, but the go-lives overlap, the measured agent writes who caught each miss, and there is no comparison arm. Recent work (August to October 2026) points the same way on benchmarks. Control is shown; influence is not. The test is turning a counterweight on and off by session, against the pre-specified null that closes the program if recurrence of the tracked error classes does not move.

Keywords: influence, sycophancy, counterweights, stopping moment, runtime guards, evaluation of agents, miss register, wiring checks

CONTENTS

1 The problem and the idea
2 The system, end to end
3 How it was built up
4 How to tell whether it works
5 What would test whether it works
6 What it means
7 Recent work and the sources
8 Limits
Appendix: scores and reasons, the latch pairing, names and receipts
References

1 The problem and the idea

An agent that works for hours decides for itself when to stop, and it stops when it can defend its answer, not when the answer is correct. This note is about what one practice, Wolfberg LLC (one operator and an AI team), built against that. It has four parts, in this order: the problem and the idea (Section 1); how the system works end to end (Section 2); how it was built up over time, and what is planned (Section 3); and only then whether it works,

what the evidence shows and where it runs out (Sections 4 to 6). Recent work and the dated sources are in Section 7.

The thesis. All you can do is influence an LLM. A second statement goes with it: what enters and leaves the LLM can be controlled, with a catch on the way in, because tool results (fetched pages, file reads) enter on the model's own choice (Section 2.1). Both are statements from a working session on 16 October, not logged to a file; Table A6 has a row for them. The models are rented and their weights are never touched, so the model is a box that cannot be opened. Control sits at the two edges of the box: at the output edge, what leaves (the Stop, refusals, the one forced retry, a verifier on the agent's own claims), and, in part, at the input edge, what enters (the context it loads, the rules, injected checks, the hooks that decide what it can reach). Influence is what that control buys inside the box. A rule written into the context is control at the input edge alone, and Section 1.2 restates exactly what the series' "0-for-7" does and does not show about how far it carried.

The idea. Counterweights are checks that are external to the agent's own output stream, that fire on actual error rather than on how confident the agent sounds, and that arrive at the moment the agent stops (program document v1.0, section 4.3). The design thesis, which is tested here and not assumed (a statement in a working session; Table A6 lists it as having no file), is plain: a check inside the loop that is answering can be talked past, and a separate process writing to a ledger the producer cannot close is a gate. The series' background paper 4 gives the same rule in its own words: an instrument the measured party operates will rot, and one that operates on the measured party will not.

1.1 The stop is the failure

From one practice. The program's thesis, stated on 30 August, is that a model presents its first answer as fact because it judges the answer defensible and expected, and that correctness is not a factor (program document v1.0, section 1.1). Conjecture: restated as a mechanism, the stopping rule is satisfiability, not correctness: a search ends when the agent can construct a defensible account from what it holds, and because more evidence can only make an account harder to defend, the incentive at the margin is to stop. The defect sits at the termination decision, not in the text the model generates before it.

Three recent measurements fit that picture without proving it. OverclaimBench measures the stop directly: a final response that reports work the agent's own transcript shows it did not do, for example reading a file it never opened. Its headline is that the report is not a reliable account of the execution (Smyth et al., 2026). That is the failure the claim guard was written for, and the benchmark's own limits are the usual ones: five scenarios, designed while iterating against one vendor's agent. XYEval reports that early inappropriate compliance is strongly correlated with task failure, and that telling the agent in its instructions to watch for it recovers only part of the loss (Wu et al., 2026). Big-Big-SycoBench separates responsiveness to valid criticism from compliance with invalid pressure and reports both, because a model that never moves is not discerning (Ballon et al., 2026). That is the same requirement as the program's H3, a reversal is not a correction, and the same reason a count of position changes under challenge cannot be read as evidence that a counterweight works.

One of the practice's own tests bears on the premise. On the full read of the operator's transcripts, the pre-registered question was whether the chance of pushback rises as context since the last reset grows. The primary fit did not converge, the verdict stayed "uninformative", and the point estimate was an odds ratio of 1.045 per doubling of context, below the smallest effect of interest set in advance (1.10); the pushback flag it rests on finds about 42% of the operator's pushback by a hand check (provisional; Table A6). Sensitivity fits give intervals of about 1.00 to 1.10 around 1.045 to 1.048 (one pre-registered sensitivity, with pressure excluded, is labeled "supported (provisional)"; a post hoc fit that drops the zero-event window gives [1.001, 1.090]), so a small

upward drift below the smallest effect of interest is not excluded. No claim is made here that an agent's agreement drifts upward within a session on this record.

1.2 Why rules fail and mechanisms are needed

Position, from one practice. A written rule that an agent has to go and read competes with the work for attention, and loses (Atkinson, 2026, background paper 3). The series' evidence for it is smaller than the shorthand suggests, and it is restated exactly here. The boot page's “0-for-7” is seven misses of one act on 12 July 2026, which the page itself counts as one act and not seven failures; the practice's corpus of at least 125 rule-bearing items has never had its adherence measured; and two tracked error classes came back in the very next session in 48 of 75 intervals, against about 40 expected from prevalence alone. What the series reports as holding is different in kind: rules loaded into the agent's context before it starts, and boundaries that refuse an action when it is attempted and record the refusal. It reports that as a mechanism, not a rate.

The repository carries the same lesson in its own comments, which are the better evidence. The command guard's header records that two of one session's eight misses were classed as repeated misses and that both rules were written verbatim in the file loaded into every coding agent before its first write, so “writing it down harder is a change that has already been tried and measured failing”. The stale-copy guard's header records an agent answering from a git mirror of a canon page that was five internal versions behind the live one, although three other things already said the mirror was stale. The design rule drawn is that an instrument the measured party operates will rot, and one that operates on the measured party will not (Atkinson, 2026, background paper 4). XYEval's partial mitigation from a system instruction is a recent result of the same general kind (an instruction recovers only part of the loss) on a different task; it is an analogy, not evidence about the command guard (Wu et al., 2026).

It follows that a prompt-level “be more critical” is not a counterweight under this specification. It is inside the agent's output stream, it asks the agent to grade itself, and it cannot be told apart from the agent's own text; it fails clause (a) under both readings given in Section 4.1. No such instruction is counted in the inventory.

2 The system, end to end

The system is described in the order a piece of work meets it: the commitment moments it passes through and the three positions a check can hold (Section 2.1); who holds the pen for each check (Sections 2.2 to 2.4); and where everything the checks do is written (Section 2.5). Code names for each check are in Table A3.

2.1 The commitment moments and the three positions

A counterweight, in the specification, is a challenge-producing mechanism satisfying three clauses: (a) external to the agent's own output stream, (b) discriminating on actual error, and (c) delivered at the stopping moment at a rate above the ambient error rate (program document v1.0, section 4.3). Section 4.1 says how each clause is scored. What matters for the shape of the system is clause (c): a check has to arrive at a commitment moment, the point at which work would otherwise pass into something that counts.

Each check is also a control point at one of the two edges of a model that is rented and cannot be opened. The edge rule, given once here and applied to every row of Table A1: a check is at the input edge if it acts on what the model receives before it acts (the context it loads, the rules, text injected into a turn, what it can read or reach), and at the output edge if it acts on what the model emits or stops on (the final text at the Stop, a tool call, a merge,

a write). Two further classes: audit, a check on the harness or the pipeline rather than on the model's input or output (the tapes, the kill check, the wiring check and the launch gate), and both, a person who writes the rules and reads the answer. The classification is a judgment and not a measurement, and Table A1 gives one reason per row.

Control at an edge does not change the model; what it buys is influence inside the box, and that is the quantity the evidence in Section 4 reads. Two limits apply. At the output edge the control lasts one turn: a refused Stop holds the output back once, and the latch means the retry cannot be blocked, so the gate's control is a forced retry, not a veto. The output-edge controls that are built are the Stop hooks and their verifier, the action guards and, as built, the operator's merge and publish go, the merge readers (spawned by the producing agent) and the canon-write evidence guard (not shown to be registered); a publish gate and ship and promote checks are planned, not built. At the input edge control is partial: tool results (fetched pages, file reads) enter on the model's own choice, and what can be governed is what the model can reach, not the content it gets back. That is the gap the planned checks are meant to narrow (Section 3.2).

Plan v0.5 (approved 7 October) names five commitment moments for mechanisms: Stop, Launch, Ship, Promote and Publish, with three board items for the three that had none (Ship, Promote and Publish). Its plan-level hypothesis numbered H3 is a different object from the program document's H3: it says a refusal buys one turn-back, and the retry is verified but cannot be blocked. The two are kept apart by always saying “the program's H3” or “the plan's H3”. Three moments the plan's five omit are added here, because mechanisms exist there: the action (a tool call), the merge, and the checkpoint during work, plus after-the-fact checks that run nightly.

Each check sits in one of three positions. The agent's own session: hooks that the harness runs inside the coding agent's session; they read the transcript or the tool call, decide by code, and can turn the agent back once. A separate process: a reviewer session, readers on pull requests, a launch gate, nightly checks and the two-pass read, which run apart from the agent that wrote the work. A person: the operator reads the answer and pushes back, merges every pull request, gives the go at launch and publish, and issues the receipt on the Promotion Ledger. The design thesis puts the weight on the second and third positions; the first is where checks are cheapest to build and, on the thesis, where they can be talked past. Placed by edge (column Edge of Table A1), ten of the fifteen mechanisms scored in Section 4.2 sit at the output edge, two at the input edge (the delivery guard, which puts text into the agent's context at the Stop, and the read, mirror and compaction gates), two audit the harness or pipeline (the launch gate, and the tapes, kill check and wiring check), and the operator works at both, writing the rules and reading the answer.

Exhibit 1 places every mechanism by position and by commitment moment, and picks out the Stop. Counts are from Section 4.

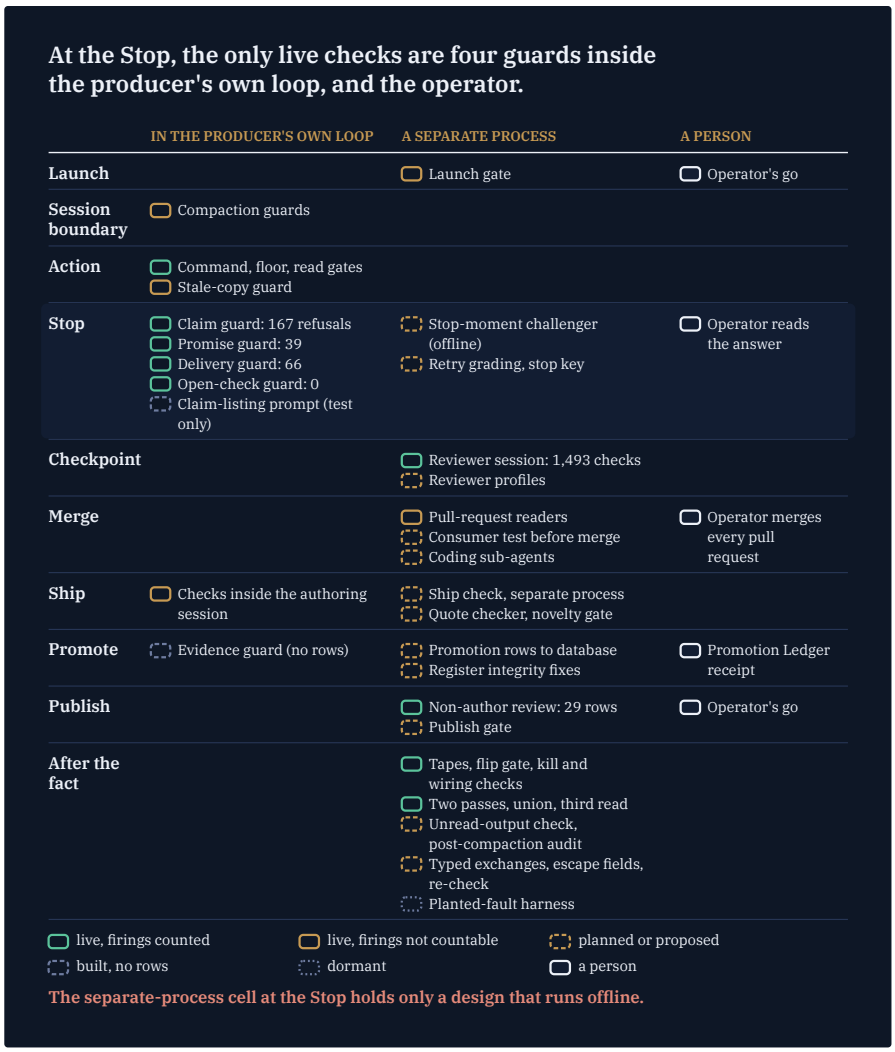


Exhibit 1. The component map. At the Stop, four hooks sit inside the producer's loop and the operator reads the answer; the cell for a separate process holds only a design that runs offline. Separate processes sit at launch, checkpoint, merge, publish and after the fact. From the hook configuration copied on 7 October, the repository, the ledgers and the board (Table 9 and Table A6). The live hook file was not readable (Section 8).

2.2 In the producer's own loop

These are scripts the harness runs inside the coding agent's session. They read the transcript or the tool call, and they decide by code, not by asking the model. Four share one design rule, a latch: the harness flag is honored, so a Stop guard turns a turn back at most once, and every internal error fails open, because a guard that can trap a session gets switched off and takes its receipt with it (claim guard header).

- **Claim guard** (Stop, first row 3 Sep). Blocks once when the final text asserts something about external state, including an absence, and no read tool call occurred anywhere in the turn. Quoted text is masked first. It also writes a stop certificate at the Stop (first on 11 September), and a verifier with no model then opens the transcript and the verify store for each claim in the certificate and writes a verdict; a false or missing pointer turns the agent back (Section 4.5). It covers the coding lane only: chat agents have no hooks, and the 157 operator turns about reading that motivated it all happened in chat.
- **Promise guard** (Stop, 3 Sep). Blocks once when a present-tense action claim (“doing X now”) ends a turn with no tool call after it, on the standard that prose is fakeable and a tool call in the same turn is not.

- **Open-check guard** (Stop, 3 Sep). Holds the turn when the agent itself registered a quantity and left it unverified; it clears only when a second reading lands in the request, not by agreement or apology. It fires only on a quantity the agent declared.
- **Delivery guard** (Stop, first refusal 1 Oct). A re-timing, not a new counterweight: when a sentence in the final text matches a matcher built from a lived experience, the agent sees the operator's verbatim correction and that experience at the Stop instead of at boot. Nothing it delivers is authored for the case. Its randomized arm exists in code, and no agent in the refusal join has been assigned to it.
- **Command guard and hard floor** (action, before a tool call). Refuse command shapes that earlier misses bought (interpreter escapes, truncated receipts, gating through a pipe), with the safe rewrite named in each denial. They emit a row only on refusal, so they have no invocation denominator.
- **Context guards.** The compaction guards act at the compaction boundary and before writes after one; the stale-copy guard denies reading a canon mirror known to be stale and names the live page; the read gate keeps agents out of the operator's override directory. Only the read gate writes rows that can be counted; the repository's own join lists four registered channels (the compaction guard, the evidence guard, the projection guard and the stale-copy guard) that wrote no row to the shared log, which makes them “unmeasured, never a clean zero” (“registered” here means entered in the channel registry, not shown to be in the hook file). The projection guard refuses a boot on a truncated projection; the research agent read only its header, it writes no row to the shared log, and it is not scored.
- **Evidence guard** (before canon writes; committed 11 September). Refuses a canon write (a board row, a ruling, a page) for a task whose evidence spec is unmet, and names each missing requirement: a file in the running tree, a test in the runner's list, a classified channel, or rows in the shared log. It is a check at the “recorded as done” step near the Ship and Promote moments. The research agent read the script and its channel entry. Its requirements come from files and registries, not from the agent, but the agent can set an escape token, and the guard fails open on any error and on any write for a task with no spec; three tasks carry specs. It writes a row for every canon write it evaluates, and there is none in either ledger. Its channel entry says it was not registered in the hook file when it landed, that registration is the operator's gate, and that it writes no rows until then. Whether it is registered now cannot be said: the hook file is unreadable to the agents. It is scored in row 7 of Section 4.2.

One built mechanism is not in force: the claim-listing prompt, the fixed text at the stop boundary that asks the agent to list each claim, the command behind it and what the command does not establish. It is absent from the hook configuration and is declared TEST-ONLY on the unwired list, untriaged since 19 September.

2.3 In a separate process

- **Pull-request reader agents** (merge). Reader agents in detached worktrees review the coding agents' pull requests and block. By the 7 October record, one pull request had three reads and two blocks and another two reads and one block; commit messages of 4 October cite the first read's five findings and the second reads of two other pull requests. They are not the pull request's author, but the producing agent decides when to spawn one and records its verdict.
- **The reviewer session** (checkpoint). A separate Claude Science session with production capability removed writes findings into a table: claim, verdict, severity, quoted evidence, a span hash, a rebuttal channel. On 16 October the table held 1,493 rows from 7 to 16 October. The research agent re-verified that none of the 807 reviewer session ids on those rows appears in the frames table that a read-only query from this project returns. That is a statement about that table, not proof that the producer cannot read the reviewer. The

producer's only move on a finding is to mark it addressed, a self-report. A warn is pull-only; a fail interrupts. The reviewer is a Claude model on every row (1,236 rows name one tier, 215 another and 42 others), the same family as the agents it reviews. It scores Partly on (a): a judge drawn from the same population of models can ratify a blind spot it shares (Rhodes, 2026), and who spawns the reviewer and who closes its findings is not shown. The retrieval asymmetry, not model diversity, is what would make it more than a vote.

- **Non-author document review** (publish). A sub-agent that did not draft the note reads the sources and grades each finding BLOCK, FIX or MINOR; the drafting agent applies every row and logs it. The 16 October review of the human-gate note returned 3 BLOCK, 18 FIX and 8 MINOR, and all 29 rows were applied or changed. The review of this note (42 rows) is logged the same way. An independent audit of 548 revision patches to the series on 7 October, made after a panel of AI reviewers' findings, amended 96 and reverted 2; version v1.1 was later superseded because it hedged or contradicted most of the series' positions. The audit's object is the revisions, not the original documents, so it is evidence about the error rate of revision patches. Plan v0.5 records 41 majors that the operator's own checks had passed in published papers, and that the review ran after publishing, which the planned publish gate is meant to reverse.
- **The launch gate** (launch). Code that blocks: a script the full-read job must call, which samples tokens from the model's usage fields, runs a pilot, shows the estimate against the budget and a pause line, and needs a written go from the operator. The go for the full read of 36,657 items is on file (15 October, 13:50). The gate used a length-fitted projection of 72.15M tokens and passed; its token-measured projection of 81.82M was over the pause line (0.85 of the 90M budget, 76.5M). The runner's own budget pause then tripped at block 60 on 15 October: the counted 75.19M plus in-flight work plus the block's own estimate of about 2.3M, scaled by 1.072, projected 78.54M, over 85% of 90M. A ruling raised the line to 0.89 (80.1M) for block 60 only and, for block 61, to a cap of 84.0M. The count at completion was 81,968,020 of the 90M budget. The mechanical pause fired and the operator overrode it twice. The gate also scores the union of two reading passes against an independent third read.
- **Two passes, the union, a third read** (after the fact). Of the 36,657 items, 33,494 were read twice and the union kept; 3,163 were exact duplicates of earlier items and were not read. For 31,490 of the 33,494 both passes ran on the main model; for 2,004, at least one pass came from a fallback utility model after the main model refused. The union recalled 98% of the records in a third read of 153 items (floor 90%) and found 8 of 8 pushback items, where the first pass alone recalled 94.1% and found 7 of 8, and the second pass alone recalled 94.8% and found 8 of 8 (eight pushback items is too few to rank the passes). The two passes disagree on 5,126 of the 33,494 items, and that list is the input to the paused second-reader bake-off.
- **Nightly checks** (after the fact). Regression tapes replay recorded failure classes as executable tests: 16 of 16 green on 16 October, with a green tape for 7 of 10 classes; the flip gate holds when a tape goes red or vanishes, but a HOLD only reports. These two rows score as firing on a pass-fail or listing output (16 of 16 tapes green; a list of 5,126 disagreements), not on a refusal. The kill check asks of each standing ruling whether what it kills still stands: 288 standing, 57 machine-checked, 51 landed, 6 orphans. The wiring check lists producers with no caller; 83 are declared exempt or untriaged.
- **A planted-fault harness** (dormant). A library of 17 labeled faults drawn from the practice's own error classes, a seeded draw whose commitment hash is published before a window opens, and scoring by type and difficulty. One window opened on 31 July, five faults drawn, and v4.10 demoted it on 2 August to “a pilot, not a measurement”. On 16 October escapes plus a random re-check were chosen over planted items for measuring the operator (a session statement; no file).

2.4 A person

The operator reads the answer and pushes back, merges every pull request, gives the go at launch and publish, and issues the receipt on the Promotion Ledger. That Notion page holds six rows, all marked PROPOSED and none promoted; four show the receipt as pending and two cite earlier rulings. It states that a change reaches production only through the ledger and that it is append-only, which the research agent could not check from the page. The operator is the only check that is independent of the producer and at the Stop, and the program's own section 1.3 calls the arrangement structurally unscalable. A working session on 16 October stated that the operator can miss and be wrong (a session statement; no file), and a board note records that the easy stuff to miss is stuff buried in code. The program reads the operator as a rater with its own severity and drift, which the companion note on the human gate (forthcoming, not yet published) takes up and this note does not repeat.

2.5 Where the events go

Every guard refusal, evaluation, stop certificate and certificate verdict is written to the session-health ledgers (a live file and an archive that pruning moves rows into). The agents and the operator log misses into the same files; a nightly producer rebuilds the public register at wolfberg.ai/register from them; the refusal join and the local-database loaders read them too. Exhibit 2 shows the path and where it breaks: the agents hold the pen on their own rows, guard rows never reach the register, the caught-by field is set by the writing agent, and no path carries an escape flag. Those breaks matter for the evidence in Section 4, and they are the reason the thesis asks for a ledger the producer cannot close.

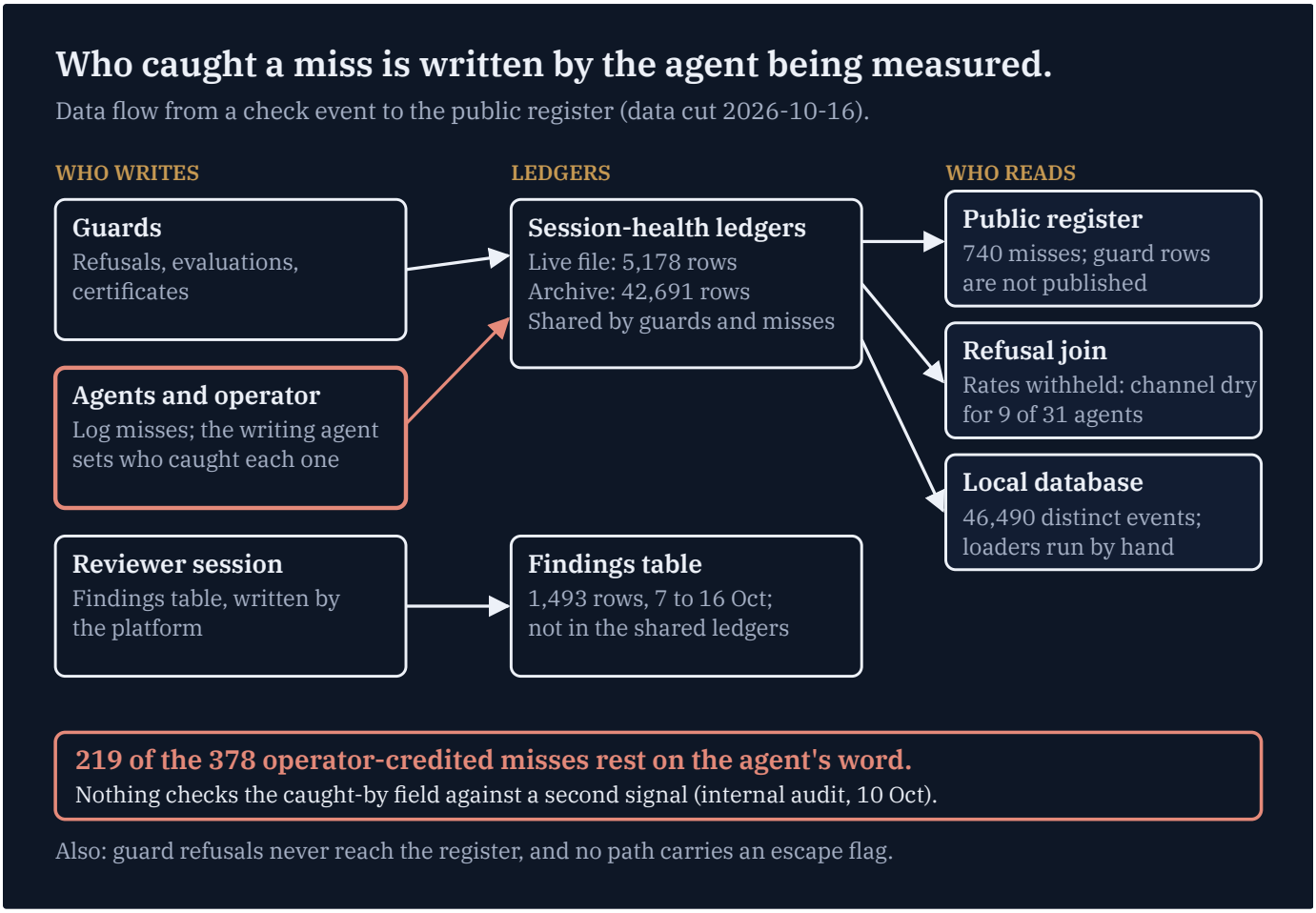


Exhibit 2. Data flow from a guard or check event to the public register. Guard rows reach the refusal join and the local database but not the register; the research agent's catches had no path to the ledger before 16 October; and no path carries an escape flag. Counts from Section 4; the path from the register audit of 10 October and the repository.

3 How it was built up

The system was not designed and then installed; it accumulated, in clumps, over four months. Section 3.1 tells the sequence, with every date. Section 3.2 lists what is built but unwired and what is planned or proposed, and Section 3.3 makes a design point about how to add the planned checks. The source of each date (commit, ledger row or board date) is in Table A4.

3.1 When each went live

Table 1 gives each go-live date; Exhibit 3 shows the sequence. The register's first row is 13 June, and a miss inbox for agents with no command line follows on 17 June. The first mechanical checks are nightly ones: the kill check (committed 17 July, UTC), which asks of each standing ruling whether what it kills still stands, and the compaction tripwire (18 July). A pilot of planted faults opened on 31 July and was demoted to a pilot on 2 August. In August came the regression tapes (16 August), the flip gate that holds when a tape goes red (20 August), and on 27 August the command guard (an action guard) and the promise guard (the first Stop guard); the program document that defines the three clauses is dated 30 August. The first rows of the session-health ledger date from 2 September, and 3 and 4 September brought the open-check guard, the claim guard, the stale-copy guard and the wiring check together. On 11 September one commit added the Stop certificate, the certificate verifier (a verifier with no model) and the evidence guard (the first certificate and verdict rows are 11 September 20:49Z; the schema last changed on 14 September and the verifier on 15 September; the evidence guard has never written a row). The hard floor followed on 17 September (first refusal 19 September), the read gate on 28 September and the delivery guard on 30 September. October added the pull-request readers (first read cited on 4 October), the reviewer session and the non-author document review (7 October), the launch gate (14 October) and the two-pass union (15 October); on 16 October the research agent entered the miss pipeline, the wiring checks were ruled and escapes plus a random re-check were chosen. Every date is a build date or a first ledger row, not a wiring date: the agents' hook registrations are in a file the agents could not read.

Table 1. The go-live date of each mechanism. Groups join events within 3 days of each other. The date is the first commit adding the file (UTC), the first ledger row where one exists, or a board close or open date. The hook registrations in the agents' settings file are not in any file the agents could read, so no date here is a wiring date; the first ledger row is the earliest proof that a hook ran. Only the last row rests on memory. Commit hashes and ledger timestamps for each row are in Table A4.

Group	Date (UTC)	Mechanism	Basis
1	17 Jul 01:15Z	Kill check	commit (build date)
1	18 Jul 05:14Z	Compaction tripwire	commit (build date)
2	16 Aug 13:52Z	First five regression tapes	commit (build date)

Group	Date (UTC)	Mechanism	Basis
3	20 Aug 01:47Z	Flip gate	commit (build date)
4	27 Aug 20:48Z	Command guard	commit (build date)
4	27 Aug 20:52Z	Promise guard	commit (build date)
4	30 Aug 12:48Z	Pre-compaction memory guard	commit (build date)
5	3 Sep 02:55Z	Open-check guard	commit (build date)
5	3 Sep 06:00Z	Claim guard	commit (build date)
5	4 Sep 09:28Z	Wiring check (nightly)	commit (build date)
5	4 Sep 16:32Z	Stale-copy guard	commit (build date)
6	11 Sep 20:22Z	Stop certificate (claim guard writes one per Stop)	commit (build date)
6	11 Sep 20:22Z	Certificate verifier (a verifier with no model, run inside the Stop hook)	commit (build date)
6	11 Sep 20:22Z	Evidence guard (before canon writes): built, no row in the shared log	commit (built; never ran)
7	17 Sep 19:08Z	Hard floor	commit (build date)
8	28 Sep 17:37Z	Read gate	commit (build date)
8	30 Sep 19:06Z	Delivery guard	commit (build date)
9	4 Oct 20:19Z	Pull-request reader agents (first read cited)	commit (build date)
9	7 Oct 03:20Z	Reviewer session	first findings row
9	7 Oct 07:22Z	Non-author document review (patch audit, two DO NOT SHIP reads)	file or page date
10	14 Oct 15:22Z	Launch gate (used 15 Oct)	board date
10	15 Oct 09:44Z	Two-pass union mode	board date

Group	Date (UTC)	Mechanism	Basis
10	16 Oct 07:30Z	Wiring checks ruled	board date
10	16 Oct 08:13Z	Research agent enters the miss pipeline	board date
10	16 Oct 09:00Z	Escape fields and 1-in-10 re-check chosen; joint ledger seed	memory only: session statement; seed file dated 16 Oct

Four events are not in the table because no mechanism went live on them: the register's first row (13 June), the first session-health ledger row (2 September, 03:57Z, refusals that carry no guard field), the arm machinery of the stop-moment challenger (built 1 September and never assigned), and the planted-fault window (31 July, a pilot demoted on 2 August). The dates from 17 July to 30 September are the build dates of scripts; for the hooks that fire only through the agents' configuration, the ledger rows of 3 September to 3 October are the first proof that they ran.

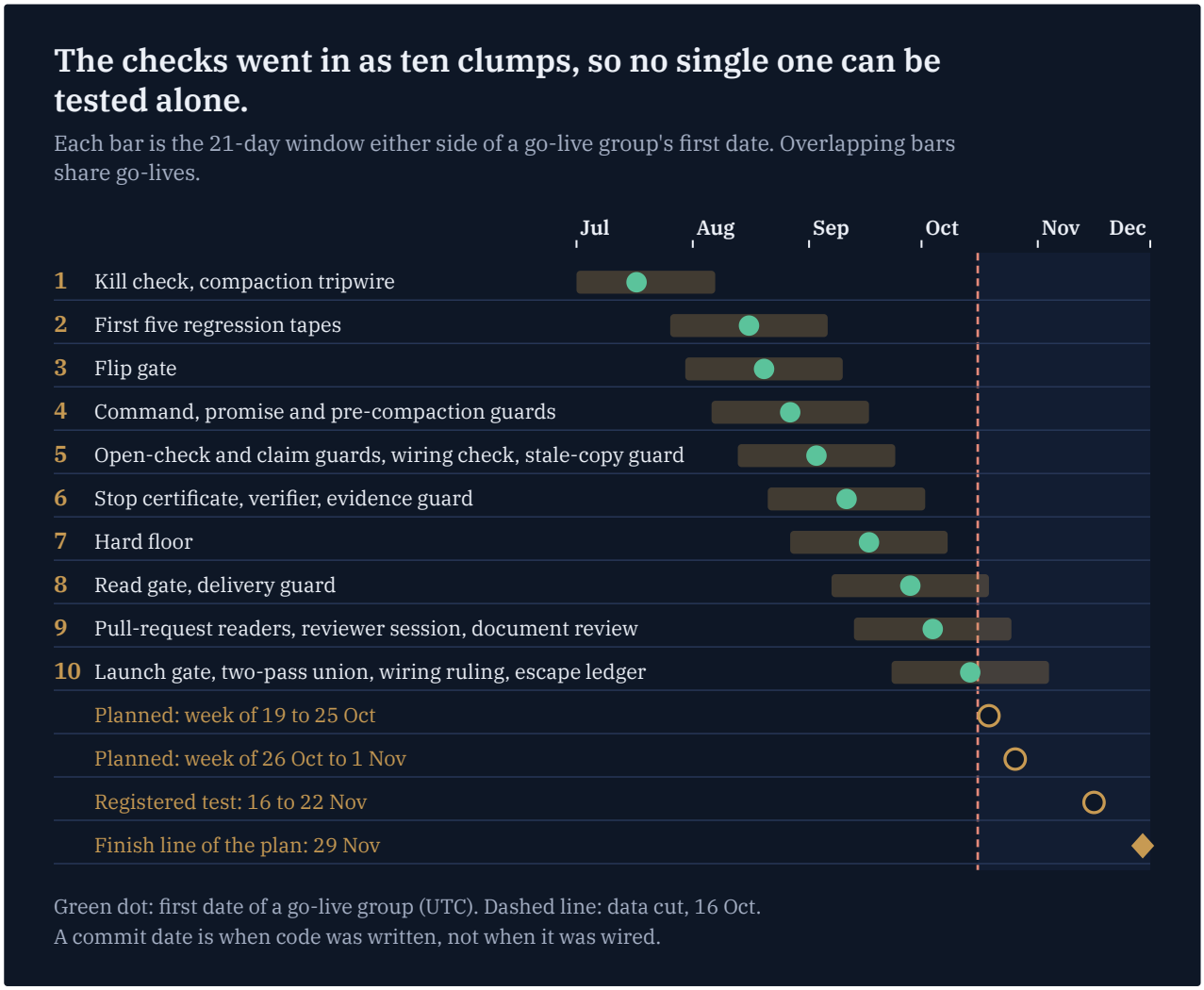


Exhibit 3. The timeline. The four Stop hooks date from 27 August to 30 September; the reviewer session, the launch gate and the two-pass read arrive in the first half of October; fourteen planned or proposed items (seventeen counting each separately) are listed in Table 2, and the board's sprint dates are marked as future points. Dates are first commits or first ledger rows (git log of the metrics repository; the ledgers), sprint dates from the board, and rulings from the

program document and the plan. This figure shows local dates; Table 1 gives the same events in UTC, which can differ by a day. A commit date is not a wiring date.

3.2 Built but not wired, proposed, and planned

- **The stop-moment challenger.** A script writes a challenge from the record or not at all: it must quote the record, cite only evidence identifiers it carries, and never speak as a persona. No live hook imports it, which matches the ratified design (clause (c) alone, with no discriminator), and in the refusal join all 17 agents read UNASSIGNED, with 0 in either arm.
- **Proposed in the research agent's records, 7 to 10 October, not built:** a quote checker that requires an exact match to a fetched source for every quotation, a novelty gate that holds any “new, first, unique” claim until a prior-art search is logged, and a post-compaction audit that marks each number used after a fold as sourced or not.
- **Planned on the board:** the ship check as a separate process, promotion rows logged into the local database, a publish step that refuses without a review receipt, the two wiring checks, and two items that harden the register itself (sealing rows before they are pruned, and moving its signing key off the agents' machine).

A question put on 16 October asked whether there are other counters still to add. There are, and Table 2 lists fourteen rows (seventeen items) with the board's status read live on 16 October: eleven unbuilt additions (some are logging or register integrity rather than counterweights), two measurement fixes for existing Stop hooks, and the stop-moment challenger, which is built but offline. None is a built, live control; the scorecard (Exhibit 4) lists them as “planned, not built” and does not score them on the clauses or the rungs, and the timeline and the event-study figure mark the board's sprint dates as future points. Of the fourteen, two are not new checks: one settles the key that defines a Stop and the other grades the retries, so they are measurement for the Stop hooks that already exist. The escape fields and re-check, and the three proposals (the quote checker, the novelty gate and the post-compaction audit), have no board item. The stop-moment challenger is built but offline only, and the Notion Promotion Ledger that the promotion-rows item would log holds six rows, all PROPOSED.

Table 2. Counterweights planned or proposed and not built. Board status read from the project board on 16 October 2026 (planned weeks are the board's sprint dates, not commitments). Board item numbers are in Table A5.

Counter	Moment	Board status	Note
Stop-moment key (settles the latch pairing)	Stop	Ready; target 11 Oct, passed	a measurement fix, not a new check
Score retries against the refused Stop (H3)	Stop	Backlog; planned week 12 to 18 Oct	grades the retry; not a new check
Ship check as a separate process	Ship	Backlog; planned week 19 to 25 Oct	separate process at Ship; not the built evidence guard, which has no rows in the shared log
Promotion Ledger rows logged into the local database	Promote	Backlog; planned week 26 Oct to 1 Nov	the Notion ledger holds 6 PROPOSED rows
Non-author review before publishing	Publish	Backlog; planned week 19 to 25 Oct	review before publish, catches flagged pre/post

Counter	Moment	Board status	Note
Reviewer profiles (non-author and researcher)	Checkpoint	Backlog; planned weeks 12 to 18 Oct (first) and 19 to 25 Oct (second)	a reviewer with its own profile
Coding sub-agents: lead, pull-request reader, follow-up	Merge	Backlog; planned week 19 to 25 Oct	builds the pull-request readers as named agents
Typed exchanges logged with raised and closed times	After	Backlog; planned week 26 Oct to 1 Nov	time to fix, a rounds-to-fix input
Wiring checks: consumer test before merge, nightly unread-output check	Merge	Backlog; planned week 26 Oct to 1 Nov; ruled 16 Oct	mechanical, not at the Stop
Escape fields and the 1-in-10 re-check	After	no board item yet; chosen 16 Oct (session statement)	the floor check on escapes
Register integrity fixes	After	Backlog; planned week 19 to 25 Oct	seals rows before pruning, other fixes
Register signing key off the agents' machine	After	Ready; planned week 12 to 18 Oct	removes the agents' hold on the key
Quote checker, novelty gate, post-compaction audit	Ship	proposed in the research agent's records; no board item	proposed only
Stop-moment challenger	Stop	built offline, no hook imports it; all 17 agents UNASSIGNED	offline only

The input-edge gap. The planned checks are weighted toward the output edge: the ship, promotion and publish gates, the retry grading and the wiring checks all act on what leaves. Input control is partial by construction, because tool results (fetched pages, file reads) enter on the model's own choice and what can be governed is what the model can reach, not the content it gets back. As the research agent reads the one-line board items, only the two profile items (which would set what a reviewer loads) and the proposed post-compaction audit act on what a model holds or can reach, and none reads the content of a tool result; the items' acceptance criteria were not read (status: reading from item titles, not a measurement). That is the gap the plan has to narrow, and it is the part of the system where influence is hardest to attribute.

3.3 A rollout order for the planned checks

Rollout design (a design point; no rollout order has been ruled). The mechanisms already live went in close together, and Section 4.7 shows what that costs: from 16 August every window contains other go-lives, so no effect can be assigned to any one of them. The planned ones can be added so that each is its own test. Switch them on one at a time with a fixed gap between them, longer than the outcome window (the 21 days used here, or the gap the registered analysis fixes), and record each go-live date from primary evidence at the moment it happens. Better still, switch each on and off by session on purpose, so that two arms run at once and the same measure (the share caught before it reaches the operator, and rounds to fix) is compared between them; the arm machinery already exists for the stop-moment challenger and has no agent assigned. The board's current sprint dates put five of the planned items in one week (19 to 25 October) and three in the next (26 October to 1 November), which would repeat the clustering; whether to keep that order is the operator's call.

4 How to tell whether it works

Only now does the note ask whether the system works, in a fixed order: what would count as working, and how each check is scored on the three clauses read two ways (Sections 4.1 and 4.2); what the files show for each check at three levels of evidence, that it fires (Sections 4.3 to 4.5), that it catches a real error (Section 4.6) and that catches moved upstream of the operator because of it (Section 4.7); and one class of failure that no check on answers can see (Section 4.8). The short answer is in the at-a-glance figure: the system fires, it has not been shown to catch, and the one measure that moves the right way cannot say which check moved it. The measures are read as two kinds. Control is what a gate did at an edge, and the ledgers count it: refusals, latched retries, the certificate verifier's verdicts. Influence is what that control bought inside the box: a real error caught, a larger share of misses caught before they reach the operator, fewer rounds to fix. Control is plentiful here; influence is the quantity that matters and the one the files cannot yet show.

All counts below were recomputed by the research agent from the files listed in Table A6 with a recomputation script (saved beside the note), except where a row says it is as written by its producer. The two session-health files are disjoint: the research agent compared 9,335 guard-type rows and found zero rows in common, so the union is the right count. Counts that came from the non-author review and that the research agent then recomputed are marked as such. The live file held 5,174 rows at the first run and 5,178 at the last rerun (the guard rows to 11:54Z are unchanged); the archive holds 42,691; the archive is the only home of guard rows before 6 October. Data cut: 2026-10-16 11:54Z. The local database's events table, loaded at 08:07Z that day, holds 961 refusals and agrees with the ledgers at that hour (the ledgers have 1,016 by 11:54Z).

4.1 What would count as working

Three rungs of evidence, none implied by the one below. A check fires if a log shows it ran and produced an output (a refusal, a verdict, a list); for the nightly checks and the two-pass read, “produced an output” means a pass-fail result or a listing, not a refusal. A check that fires has exercised control; the two higher rungs ask whether that control bought influence. It catches if something shows a flagged item was a real error. It moves catches upstream if, after it went live, a larger share of misses was caught before they reach the operator, by the agent itself, a peer instance or an instrument; this replaces “shipped errors fell”, which a ruling of 16 October set aside as the wrong headline measure. The clause scores use the rubric in Table 3.

The specification is in the program document v1.0, section 4.3: a counterweight is a challenge-producing mechanism satisfying three clauses, each tied to a warrant in the document's section 2. The document grades its own citations (READ for full text, LEAD for abstract only, ASSERTED for secondary), and says nothing graded LEAD or ASSERTED may reach an external surface before its primary is read. The document's primaries were not read for this note. Table 3 therefore gives the document's section references, names a source only where Table 9 or Table A6 shows how it was reached, and repeats no figure from an unread source as established.

Table 3. The clauses, their warrants by the program document's section, and how this note scores each. A row is scored only on what a file shows. The one-line reason for every score is in Table A1.

Clause	Requirement and warrant	Scored here as

Clause	Requirement and warrant	Scored here as
(a) External	External to the agent's own output stream. Warrant: the document's section 2.1, on self-correction without external feedback. The primary it cites was not read.	**Strict reading (independent of the producer). Meets:** a separate process the producer cannot read, edit or close, or a person. **Partly:** code outside the model but inside the producer's session or repository; spawned or recorded by the producer; or the same model family with the spawn and the close not shown. **Does not:** the model's own text or memory. **Literal reading (the document's wording). Meets:** a check that decides by code or by another party, not the model's own text. **Partly:** a further pass by the same model. **Does not:** the model's own text.
(b) Discriminating	Fires on actual error, not on assertion confidence. Warrant: the document's section 2.2 and the role-dependent capitulation result of Kelley and Riedl (arXiv:2603.00024, 3 February 2026), which the document grades READ; neither primary was read.	**Meets:** a checkable condition and a measured discrimination. **Partly:** a checkable condition on the transcript or state beyond a sentence pattern, discrimination not measured. **Does not:** the sentence pattern is the only trigger. **Not assessed:** no basis in the files.
(c) At the moment	Delivered at the stopping moment at a rate above the ambient error rate. Warrant, offered as an analogy: the document (section 2.3) reports a preregistered crossover against a natural error rate of about 10%, in which injected signals at 1% harmed detection by 4 points, 5% did nothing and 15% helped by 7. The study, Fang et al. (SSRN 7108438), injected errors into PharmD reviewers' review of AI-drafted summaries; its public abstract reports a 6-point drop at low and a 9-point rise at high injection frequency, so the document's figures and the abstract's differ, and which version the document used is not known. Neither is a test of a challenge at an agent's stop.	**The document's moment: the Stop. Meets:** runs at the Stop and has fired there. **Partly:** runs at the Stop and has never fired, or sits at another commitment moment it is built to gate (action, canon write, launch, merge, publish, promote). **Does not:** not at a commitment moment (a checkpoint during work, or after the fact). The rate half of the clause, above the ambient error rate, is not assessed for any row: the 16 October dose join ran without reading the ambient (status: not read).

The program document's clause (a) reads “External to the agent's own output stream”. Read literally, a script that reads the transcript and decides by code is external, so all four Stop hooks score Meets on (a); on (c), the claim guard, the promise guard and the delivery guard score Meets and the open-check guard scores Partly because it has never fired. On the literal reading a separate reviewer session (row 9) and a sub-agent document review (row 10) count as “another party” even though they are the same model family, because they are separate runs whose output the producer does not write; a second reading pass by the same model (row 12) is scored Partly because it is the same labeling task done again by the same model rather than a check on the first pass's output. The rubric adds a second requirement the document does not state: the producer cannot read, edit or close the checker. The structural finding, that no mechanical check is both independent of the producer and at the Stop, holds under this stricter reading only. Under the literal reading the Stop hooks are external and at the Stop, and the open question is (b). The headline names the reading it uses and gives the other beside it; Exhibit 4 carries both.

Three further parts of the specification shape the scoring. H3, a reversal is not a correction, says a counterweight that raises reversals while leaving correctness flat is a negative result, not a partial success, so reversal counts are never read as evidence. The primary test has a pre-specified null: if the compound recurrence half-life of the tracked error classes does not move off its pre-computed baseline with an interval that excludes zero, the program closes (section 7.1). And the document requires a fidelity check, since a null cannot distinguish a

counterweight that does not work from one never delivered as specified, including a falsifier that 20 renders with zero read-after-render means the delivery mechanism failed (section 7.2); that falsifier was not evaluated here.

4.2 The scorecard

Exhibit 4 scores each of the fifteen against the clauses of Table 3. Two cautions come first. The clause scores are a judgment applied to a stated rubric, and a reader could move some of them; the judgment calls are listed in Section 8. “Fires” means a log shows the check ran and produced an output. The evidence rungs are different in kind: they ask only whether a file shows something, and a mechanism with no row in any file scores empty however good it may be. The evidence rungs for all fifteen are summarized in the at-a-glance figure, and the counts behind each row are in Table A1.

The fifteenth row, the evidence guard (row 7), was added after the first review round. The research agent read its script and its channel entry (Section 2.2) and found no row from it in either ledger. It scores Partly on (a) strict, as the other hooks do, Partly on (b), because it checks a stated condition on files and registries for the three tasks that carry specs, and Partly on (c), because it gates the canon write, a moment it is built to gate, and not the Stop. Whether it is registered in the hook file is unknown, because the file is unreadable to the agents, and the channel entry says it writes no rows until the operator registers it; what the files show is what is scored, so every evidence rung is empty.

No mechanism meets all three clauses, because none clearly meets (b), discriminating on real error.

		(a) strict	(a) literal	(b)	(c)
IN THE PRODUCER'S OWN LOOP					
1 Claim guard	Stop	◐	●	◐	●
2 Promise guard	Stop	◐	●	◐	●
3 Open-check guard	Stop	◐	●	◐	◐
4 Delivery guard	Stop	◐	●	⊘	●
5 Command guard, hard floor	Action	◐	●	◐	◐
6 Compaction, stale-copy, read gates	Action	◐	●	◐	◐
7 Evidence guard	Canon write	◐	●	◐	◐
A SEPARATE PROCESS					
8 Pull-request readers	Merge	◐	●	◐	◐
9 Reviewer session	Checkpoint	◐	●	◐	⊘
10 Non-author document review	Publish	◐	●	◐	◐
11 Launch gate	Launch	◐	●	◐	◐
12 Two passes, union, third read	After	◐	◐	◐	⊘
13 Tapes, kill check, wiring check	After	◐	●	◐	⊘
A PERSON					
14 Promotion Ledger with receipt	Promote	◐	●	◌	◐
15 The operator	All	●	●	◐	●

● meets ◐ partly ⊘ does not ◌ not assessed

(a) external to the producer (b) fires on real error (c) arrives at the Stop

Meeting (a) and (c): 1 on the strict reading (the operator), 4 on the literal reading.

Exhibit 4. The scorecard. Clause (a) is scored on the strict reading and on the literal reading, and (c) on the moment alone. No row meets all three clauses, because none clearly meets (b). The clause scores in loop rows are limited by (a) on the strict reading, because the hook shares the producer's session and the producer writes its own telemetry; the scores in separate-process rows are limited by (c), because they act at checkpoints, merge or after the fact. Clause scores are a reading of the rubric in Table 3, with a reason for each row in Table A1.

Position. The fleet holds the two halves of the specification in different places, and how (a) is read decides how wide the gap is. Read strictly, as independent of the producer, the Stop hooks have (c) by construction, since they fire at the Stop and do not have to estimate it, and they cannot have (a) without ceasing to be hooks. The reviewer session has most of the independence (a non-author role, with production capability removed) but only partly (the same model family; who spawns it and closes its findings is not shown), and cannot have (c), because it acts at checkpoints. Read literally, as code outside the model, the Stop hooks have (a) and (c), and what is missing is (b): no file shows that any of them tells a real error from a harmless one. The program's H2 says timing is the control, so a late external ledger is a different instrument from an early internal one, not a weaker copy. Nothing in the program currently varies the two independently, because the arm that would do so is unassigned.

The weakest column is (b). No row clearly meets it. The strongest case is the launch gate, which compares measured counts to thresholds. Its budget gate passed on a length-fitted projection while its token-measured projection was over the pause line; the length-fitted forecast ran 1.24 times low over blocks 1 to 59, and the whole-read count of 81.97 million tokens is 1.14 times the 72.15 million the gate passed on; the running count, not the gate, exposed that, and the joint ledger holds it as an escape found by an external instrument about a day later. Its recall gate (union 0.98 on 153 items against a floor of 0.90) is a measured comparison with a third read, a different object from the budget gate. The launch gate scores Partly, because neither has a measured false-pass or false-block rate. The remaining rows fire on a checkable condition by design and have no measured discrimination. One rule separates the three Stop hooks that fire on a sentence. The claim guard (no read tool call in the turn), the promise guard (no tool call after the action claim) and the open-check guard (a store the agent registered) each pair a trigger with a checkable condition on the transcript or the store, and score Partly. The delivery guard delivers on a sentence match alone, whatever the turn did, and scores Does not. The rule is a judgment and the split is not a measured result; the refusal join gives no basis to separate them (2 of 38 and 0 of 65 firings met a correction, rates withheld).

4.3 What fires: control exercised

Table 4. The guards that write rows, recomputed from the live and archive ledgers to 2026-10-16 11:54Z. “Stops” counts a guard's runs at the Stop: clean, latched and refused (for the open-check guard, also runs with no store registered). Guards that emit only on refusal have no run count.

Guard	First row	Stops	Refusals	Refusals per Stop
Claim guard	3 Sep	1,704	167	9.8%
Promise guard	3 Sep	1,704	39	2.3%
Open-check guard	3 Sep	1,704 (1,058 had no store registered, 386 clean, 260 retry)	0	0% of 1,704; 0 of the 386 where a store existed
Delivery guard	1 Oct	674	66	9.8%
Command guard	3 Sep	not logged	708	n/a

Guard	First row	Stops	Refusals	Refusals per Stop
Hard floor	19 Sep	not logged	8	n/a
Read gate	3 Oct	not logged	2	n/a
No guard field (2 to 3 Sep)	2 Sep	not logged	26	n/a
All refusals			1,016	

Of the 1,016 refusals, 272 were at the Stop, from 21 of the 37 sessions in the ledgers (a session here is a session id; the ledgers' session and agent ids are one to one). These are firings, and a firing is not a catch. The shares are of Stops the guards ran, not of Stops that held an error, because no file records how many Stops held one. The cost: about one Stop in ten is turned back once (claim guard 9.8%, delivery guard 9.8% of Stops), and 904 refusals had no logged correction, an upper bound on nuisance because the correction channel is dry. For the open-check guard, 0 refusals in 1,704 runs is mostly an absence of opportunity: 62% of its runs had no registered quantity to check. Four registered channels (the compaction guard, the evidence guard, the projection guard and the stale-copy guard) wrote nothing to the shared log and are absent from the table (the evidence guard is scored in row 7 with every evidence rung empty); the refusal join calls them “unmeasured, never a clean zero”.

4.4 The latch (control)

After a Stop refusal the harness runs the retry with the latch flag set, and the guards log it as latched: they cannot block it. There are 260 latched retry Stops against 272 Stop refusals. By design a retry after a refusal cannot be blocked, and it ships whatever it ships. The counts are close but not equal: the 272 refusals sit at 272 distinct anchors against 260 latched retry Stops (the reviewer's recount agrees), and under the same-guard pairing one promise-guard refusal's next row was not latched. The plan's H3 asks how often the retry repeats the failure, and that grading (about 135 retries when it was written, 260 now) has not been run. The count says only that the retry was unblockable, not that it failed.

A per-guard latch spread reported on 7 October is a property of how a refusal is paired with the next row, not of the guards (Appendix, Table A2). Settling the stop-moment key is a measurement item still marked Ready; until it is settled, no per-guard latch rate should be quoted.

4.5 What the certificates and the verifier do (control)

The claim guard writes a stop certificate at the Stops it evaluates from 11 September: 1,402 in all. Of these, 1,046 (74.6%) list no claim, with the rule label NO-CLAIMS. They are not all Stops where nothing was asserted. Joined to the guard rows by session and anchor (the research agent's recount, matching the reviewer's): 781 sit at Stops that the claim guard passed (781 of the 1,032 certificates at passed Stops, 75.7%); 162 sit at retry Stops, where the guards stand down (162 of 227); and 102 sit at Stops where the claim guard itself refused because the final text asserted something about external state (rule labels such as LANDED STATE, ABSENCE and EXTERNAL COUNT), so the certificate's claim list and the refusal detector disagree on those Stops; 1 is unmatched. The 7 October figure was 74.5%. The ledger cannot say whether any of the 781 carried claims the hook does not detect. At a passed Stop, a zero-claim certificate is a pass with nothing checked, and plan v0.5 already says to count such passes as non-checks. At a refused Stop it is an inconsistency to resolve. The rest: 356 certificates list at least one claim (282 list exactly one).

The verifier. The certificate verifier was committed in the same commit as the certificate (11 September; the schema last changed on 14 September and the verifier on 15 September). It is a verifier with no model: for each claim in the certificate it re-opens this session's transcript and the verify store and checks that the pointer exists and is what it says, and its own tests walk its import graph to check that no model is called. It runs inside the claim guard's Stop hook, on the certificate as it landed in the shared log, and writes one certificate verdict row per Stop: 1,402 rows at the cut: 1,375 pass (1,046 of them on certificates with no claim, so nothing to open) and 27 refused, all 27 on the 356 certificates that list a claim (7.6%; rule labels: test run 13, no pointer 11, transcript offset 3). Of the 27, 22 turned the agent back, each as one of the claim guard's 167 refusals (13.2% of them carry CERTIFICATE-REFUSED, 20 with no text rule alongside), and 5 fell at latched retry Stops, where nothing can block. Separately, 14 of the certificates themselves carry the verdict refused, which is the builder's own mark that a claim lacks a pointer or a write failed. These are control exercised. The files do not say whether any of the 27 was a real error; the verifier's builders replayed it on real transcripts before merge and fixed three false refusals (15 September), and the research agent did not rerun that. It also means the 1,046 zero-claim certificates (74.6% of all) are Stops where the verifier had nothing to open; they are not Stops where it did not run. It sees whether a pointer holds, not whether a read was understood, and it runs inside the process that built the certificate (Section 4.1 scores that as (a) Partly on the strict reading).

4.6 What catches (influence, first rung)

This is the first rung that reads influence and not control. The repository's own join of refusals to the operator's corrections (run 16 October 03:38Z, read here and not rerun) is the closest thing to a measurement of discrimination, and it withholds every rate. It joins by agent and a 20-minute window, not by turn. A refusal followed by a logged correction is a true positive; a refusal with none is a false positive, which cannot distinguish nuisance from prevention; an exposed correction that no guard refused is a false negative. The correction channel is dry for 9 of the 31 agents in the join, and those 9 agents hold 776 of the 950 refusals (81.7%) and 43 of the 77 exposed corrections, so a rate would be a number about logging. The join also ran without reading the ambient error rate (status: not read), which is why the rate half of clause (c) cannot be assessed.

Table 5. The refusal join as written by its producer, 16 October 03:38Z. Counts only; every rate is withheld by the instrument. Firings (950) and TP plus FP (925) differ by 25 because the instrument excludes suppressed refusals and undelivered overrides from the labeled counts. Per-guard TP counts refusals that coincided with a correction; the All row counts distinct corrections (21), so the column does not sum (50). The instrument's TP+FP can exceed firings (claim guard 164 vs 161, command guard 677 vs 676).

Guard	Firings	Coincided with a correction (TP)	No logged correction followed (FP)
Claim guard	161	6	158
Promise guard	38	2	36
Delivery guard	65	0	65
Command guard	676	41	636
Hard floor, read gate	10	1	9
All guards	950	21	904

Alongside those, 473 corrections by the operator were logged, 77 of them in an agent with a guard live, and 56 of the 77 had no refusal before them. The September window reported in the series (93 refusals in 674 invocations, 2 coinciding with a correction, 14 corrections on a turn where a guard was live and silent) had the same shape

and the same refusal to compute rates. The firing counts grew by an order of magnitude; the ability to say whether they discriminate did not.

The evidence of catching is stronger where the check sits later and is a person or a separate reader. It is scored as indirect, because in each case the drafting agent accepted the finding and no one scored it independently:

- **Reviews of documents.** All 29 rows of the 16 October review were applied or changed, three of them BLOCK; the 7 October patch audit amended or reverted 98 of 548 patches (17.9%), which is evidence about the error rate of revision patches, not about the original documents.
- **The reviewer session.** 42 fail verdicts among 1,493 rows (2.8%), 37 resolved and 5 unaddressed; 189 warn. How many of the 42 are real errors is not known. In the 7 October specimen two of seven fails were false positives on evidence the reviewer could not reach, and the table has no state for “the finding was wrong”, so an ignored real error and a correct rebuttal both end as unaddressed.
- **Pull-request readers.** Blocks followed by fixes in two pull requests (7 October record, not re-read here).
- **An offline auditor, never deployed.** In the series' test of a small local model as auditor of 170 known claims, 41 of them false, the first prompt caught none of the false ones, and a second prompt flagged 68% of the false claims and 66% of the true ones (Atkinson, 2026, background paper 4). A check can fire at the right moment and discriminate at chance.

Table 6. The joint miss ledger (operator and research agent), 16 rows from one session (16 October). An illustration of the fields, not a rate. Item ids are in Table A6.

Measure	Count
Origin: research agent alone / both / operator alone	15 / 1 / 0
Caught by: peer instance / operator / agent itself / external instrument	9 / 5 / 1 / 1
In front of the operator	10 of 16
In front of the operator and not caught by the operator	6: five caught by a peer, one by an instrument; five before ship, one after
Caught before ship / after ship	13 / 3

The shape is what matters: ten of sixteen items passed in front of the operator, and in six of those something other than the operator caught it. One session cannot say how often that happens. It does show what the register cannot: it has no field for an escape, for before or after ship, or for a random re-check sample.

4.7 A dated event study: did catches move upstream? (influence)

A ruling of 16 October, made in a session statement (no file), set shipped errors aside as the wrong headline measure for this practice. The research agent's proposal, which this note adopts, is three measures. The main one is the share of misses caught before they reach the operator: caught by the agent itself, a peer instance or an instrument, against those the operator catches (the register's caught-by field; the register's “operator” is the human operator). The second is rounds to fix: the further real pushbacks per real pushback, the secondary measure of the pushback analysis, from the operator's messages to 7 October. The third is escapes past both, kept as the floor check and not as a headline; only the 16-row joint ledger exists for it (Section 4.6). “Shipped

errors fell" remains only as the original wording of plan v0.5's Goal 3 and of the registered-test item; plan v0.5 is unchanged and waits on the operator's yes.

Design, fixed before the outcomes were read. The window is 21 days each side of a go-live. Go-lives within 3 days of each other form one joint group, pivoted at the group's earliest date. The outcome is the upstream share. The primary analysis excludes the two bulk-logged batches, because their timestamps are when the rows were filed, not when the catch happened: 77 rows stamped in one minute on 15 July (55 operator, 18 self, 4 peer) and 100 rows filed on 3 and 4 October in a retrospective sweep (73 self, 24 operator, 3 peer); a sensitivity analysis keeps them. Each side gets a Wilson interval; the difference gets a percentile bootstrap interval. Where go-lives overlap inside a window they are one joint effect and cannot be separated. Nothing here is adjusted for multiplicity.

Table 1 (Section 3.1) gives each go-live date; Exhibit 5 plots the share by month against the go-live dates, and Exhibit A1 in the appendix shows the weekly counts and the two bulk-logged batches.

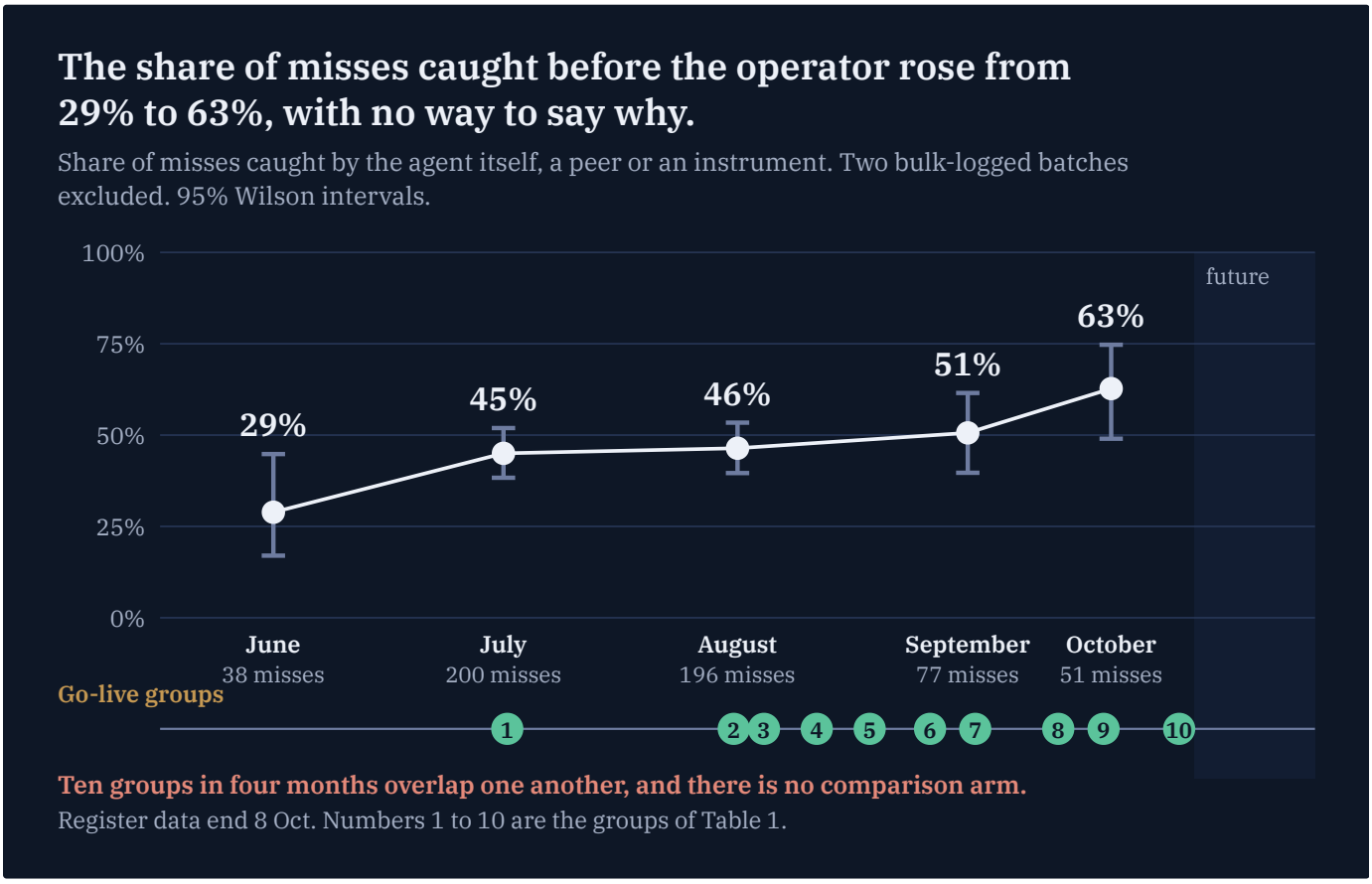


Exhibit 5. The share of misses caught before the operator rose from 29% to 63%, with no way to say why. Monthly upstream share with 95% Wilson intervals, the two bulk-logged batches excluded; the ten joint go-live groups of Table 1 are marked beneath, and the shaded area is the future. Register to 8 October. Descriptive only.

What the weekly series shows. Excluding the two bulk batches, the upstream share is 29% of 38 misses in June, 45% in July (200), 46% in August (196), 51% in September (77) and 63% in October (51, to 8 October); the monthly Wilson intervals are 10 to 25 points wide (Table 8). The October end is inflated by the sweep: with every row October reads 72% upstream, and 73 of the 100 swept rows are self-caught. The second measure shows no trend overall: the weekly mean rounds per pushback sit between about 0.9 and 2.2 from July to mid-September,

and only 2 pushbacks fall after 21 September because the measure needs five further messages read and the messages stop on 7 October. It moves in one window only, the flip gate, and that move cannot be attributed (below).

Table 7. Before and after each joint go-live, 21 days each side (fixed before any outcome was looked at). Upstream share = misses caught by the agent itself, a peer or an instrument, over all attributed register misses; the two bulk-logged batches are excluded. The difference is in percentage points, after minus before, with a 95% percentile bootstrap interval (10,000 resamples, seed 20261016). Rounds to fix: mean further real pushbacks per real pushback (the operator's messages to 7 Oct), with the number of pushbacks. “Other groups” counts the other go-live groups whose date falls inside the window. Nine comparisons, no multiplicity adjustment.

Group	Go-live (pivot date)	Upstream share before	after	Difference [95% interval]	Rounds to fix before to after (n)	Other groups	Flags
1	Kill check; Compaction tripwire (2026-07-17)	57/141 = 40.4%	43/87 = 49.4%	+9.0 [-4.3, +22.2]	1.442 (120) to 1.27 (37)	0	77-row July stamp in the before window (excluded)
2	First five regression tapes (2026-08-16)	18/50 = 36.0%	83/181 = 45.9%	+9.9 [-5.5, +25.0]	1.75 (32) to 1.737 (198)	3	none
3	Flip gate (2026-08-20)	29/83 = 34.9%	72/148 = 48.6%	+13.7 [+0.4, +26.3]	1.974 (115) to 1.522 (113)	3	none
4	Command guard; promise guard; pre-compaction memory guard (2026-08-27)	72/154 = 46.8%	41/87 = 47.1%	+0.4 [-12.7, +13.6]	1.822 (174) to 1.766 (64)	5	none
5	Open-check guard; claim guard; wiring check (nightly); stale-copy guard (2026-09-03)	83/180 = 46.1%	35/64 = 54.7%	+8.6 [-5.5, +22.6]	1.797 (187) to 1.78 (50)	5	none
6	Stop certificate, certificate verifier; evidence guard built the same day, never ran (2026-09-11)	68/131 = 51.9%	42/64 = 65.6%	+13.7 [-0.9, +28.3]	1.522 (90) to 2.045 (22)	4	research agents begin in window
7	Hard floor (2026-09-17)	41/88 = 46.6%	45/72 = 62.5%	+15.9 [+0.4, +30.9]	1.766 (64) to 0.333 (6)	5	after window 20 of 21 days; 100-row sweep in window (excluded); research agents begin in window; rounds thin
8	Read gate; delivery guard (2026-09-28)	32/50 = 64.0%	33/53 = 62.3%	-1.7 [-20.8, +17.4]	2.045 (22) to 0.0 (2)	4	after window 10 of 21 days; 100-row sweep in window (excluded); research agents begin in window; rounds thin

Group	Go-live (pivot date)	Upstream share before	after	Difference [95% interval]	Rounds to fix before to after (n)	Other groups	Flags
9	Pull-request reader agents (first read cited); reviewer session; Non-author document review (patch audit, two DO NOT SHIP reads) (2026-10-04)	32/50 = 64.0%	18/33 = 54.5%	-9.4 [-31.5, +11.7]	1.556 (9) to 0.0 (1)	3	after window 3 of 21 days; 100-row sweep in window (excluded); research agents begin in window; rounds thin
10	Launch gate (used 15 Oct); two-pass union mode; wiring checks ruled; research agent enters the miss pipeline; Escape fields and 1-in-10 re-check chosen; joint ledger seed (2026-10-14)	35/60 = 58.3%	0/0 = n/a	n/a [n/a, n/a]	0.0 (2) to n/a (0)	2	after window 0 of 21 days; 100-row sweep in window (excluded); research agents begin in window; thin (fewer than 20 rows a side); rounds thin

What the windows show. Nine of the ten groups have any after-window; six have the full 21 days (groups 1 to 6) and group 7 has 20. For those seven the upstream share is higher after than before, by +0.4 to +15.9 points, and group 4 (+0.4) is the only point estimate near zero. Two intervals exclude zero: the flip gate (group 3, +13.7 points, [+0.4, +26.3]) and the hard floor (group 7, +15.9, [+0.4, +30.9], 20 days after). With nine comparisons at 95% about half of one would be expected to exclude zero by chance, and both of these windows contain three to five other go-lives. Groups 8 and 9 have 10 and 3 days after them, and group 10 has none. Rounds to fix moves in only one window, the flip gate (group 3: 1.974 rounds before to 1.522 after, interval [-0.79, -0.11]); that window contains three other go-lives (groups 2, 4 and 5), so the move cannot be attributed to the flip gate. The windows from group 6 on have fewer than 30 pushbacks after. Keeping the two bulk batches changes the group 1 interval to [+0.8, +25.4] (the July batch lies in its before window) and the group 7 interval to [+11.2, +35.9] (the October sweep lies in its after window), which is why the primary analysis excludes them.

Why this is not an effect. Group 1, the kill check and the compaction tripwire, is the only go-live with no other in its window, and its interval includes no change. From 16 August every window except the last contains three to five other go-lives, so none isolates one mechanism, and the last window has no days after it and cannot be read either; the study can say only that the upstream share rose over the period in which the checks were added. Other things moved it too. Who sets the caught-by field moved: the writing agent sets it, the self-catch count depends on how readily agents log their own catches, and the research agents entered in early October. And the logging itself changed in the two bulk batches. There are 17 weeks, ten joint groups, logging that changed under the series, and no comparison arm. This is descriptive. The real test is to turn a counterweight on and off by session, on purpose, and compare the same measure in the two arms; Section 3.3 says how to stage the planned checks so that each one is its own test.

Table 8. Register entries by month, from the register build of 15 October 08:53Z (recomputed; 739 misses with a caught-by value, 13 June to 8 October; one more is unattributed). The ledgers match the total (741 deduped rows, the 741st dated 16 October) but not row for row: the reviewer found about 140 register timestamps that differ from their ledger rows (a batch stamped 18 July 04:46 UTC in the ledger), and by type the ledger has 178 repeated-miss rows against 177 in the register while the new row is a would-ship-error. Months are not exposure-normalized.

Month (2026)	Misses, all rows	Operator share, all rows	Misses, bulk batches excluded	Operator share, excluded	Upstream share, excluded (Wilson 95%)
June	38	71.1%	38	71.1%	28.9% (17.0% to 44.8%)

Month (2026)	Misses, all rows	Operator share, all rows	Misses, bulk batches excluded	Operator share, excluded	Upstream share, excluded (Wilson 95%)
July	277	59.6%	200	55.0%	45.0% (38.3% to 51.9%)
August	196	53.6%	196	53.6%	46.4% (39.6% to 53.4%)
September	77	49.4%	77	49.4%	50.6% (39.7% to 61.5%)
October	151	28.5%	51	37.3%	62.7% (49.0% to 74.7%)

The register holds 740 misses, of which 378 are operator-credited, 291 self, 51 peer instance, 19 external instrument (all probe misses) and 1 unattributed; all 740 are hash-chained and none is signed, and its five “gate” rows are a fixed example list. The 100-row sweep of 3 and 4 October (73 self, 24 operator, 3 peer; the 10 October internal audit, here recounted from the register's timestamps) is one of the two bulk batches. The ledgers hold one new miss row after the register's last (8 October 18:32Z), dated 16 October and typed would-ship-error, so 740 plus 1 is a count match, not a row-for-row reproduction; the correction channel went quiet while a joint ledger logged sixteen items in a session, which is the “dry channel” the refusal join keeps naming.

4.8 The class no check on answers can see: output nobody reads

A board note of 16 October records that the easy stuff to miss is stuff buried in code, and the example given was one that reading a response could not catch: components built and never connected until someone tests after merge. The files give instances from the last two weeks. The research agent's miss-drop path wrote to a folder that no importer read; one drop sat 9 days and 7 more were added the same day (joint ledger), until the ingest change merged early on 16 October. The wiring check's unwired list declares 83 producers without a scheduled caller; 42 carry a dated reason and 41 are grandfathered and untriaged (21 UNWIRED, 20 TEST-ONLY), among them the claim-listing prompt. Four registered guards write no row, so the join that measures guards cannot see them. The local-database loaders run by hand and nothing schedules them. Exhibit 2 shows the path and where it breaks.

Two mechanical checks were ruled for this class on 16 October (sprint of 26 October to 1 November). A pull request that adds a producer (a file, folder, field or script that writes output) must name its consumer and carry a test that runs through it, with CI failing if none is named. And a nightly check flags any output written more than three days ago that no registered reader has read. On whether such misses need a separate “unwired” type, the operator, 16 October 2026: “meh, not sure we really need it as its own type”. Such misses therefore use the existing types with an “unwired” note. Neither check exists yet. Both are mechanical and external, and both sit at merge or after the fact, not at the Stop.

5 What would test whether it works

Nothing below is running. These are the tests the program has already committed to or that the evidence above makes necessary, in the order they would be run.

5.1 The registered test (Goal 3)

The board item, as written, asks whether the counterweights reduce shipped errors, using register would-ship and post-publication errors per active hour, before and after the hooks, together with the program's H3 and a

non-author read. A ruling of 16 October set shipped errors aside as the wrong headline measure for this practice (a session statement; no file). The measures proposed instead are, in order, the share of misses caught before they reach the operator, rounds to fix, and escapes past both as the floor check; the board item and plan v0.5 keep their “shipped errors” wording until the operator says yes. The test is registered first, with the before and after periods stated, and it depends on three other board items (the retry grading, the ingest of the research agent's catches, and typed exchanges with raised and closed times). Three design problems are already visible in the files. First, the mechanisms went live in ten joint groups between 17 July and 16 October, nine of them from 16 August onward and each with another inside three weeks (Table 1), and the register's catchers changed over the same months (Section 4.7), so a single before-and-after split is confounded twice. The program document's own answer is a single-case design with at least three demonstrations of effect at three different points in time, and it records that its phase criteria ask for two when the standard needs three. Second, the comparator is not stationary: the hub page reads the correction base rate in the operator's chats as rising from 4.0% to 19.7% across April to August and in code from 20.3% to 25.3% across July to September. It states separately that the codebook change of 11 September roughly halved both rates (chat from 33.0% to 16.5%, code from 40.1% to 22.3%), and it does not say whether the rising series is read before or after that change, so which codebook the rising series uses cannot be said. The page concludes that a pooled figure registered as a known comparator and compared with a forward window would credit the counterweight with the drift, so the arms must be drawn concurrently. Third, the outcome proxy is a median gap that cannot move while most intervals sit at the floor of one session; the series reports 48 of 75 intervals at the floor against about 40 expected from prevalence alone (an 11 September render), and the floor share, read against that baseline, is the quantity that can move.

5.2 H3: grading the retries

The retry-grading item grades each retry's certificate verdict against the failure the refused Stop cited: fixed, said UNVERIFIED, or same failure. There are 260 retry Stops in the ledgers now. The test is to be registered before the results are read, and it needs the stop-moment key first (Section 4.4). Under the program's H3 a retry that changes position is not thereby a correction, so the grading needs an independent correctness signal; the per-claim reviewer table is the nearest candidate and is same-family.

5.3 The arms, and the null that closes the program

The interventional arm cannot start until agents are assigned: in the refusal join, 17 agents read UNASSIGNED and none reads ON or OFF. The pre-specified rule is that the compound recurrence half-life of the tracked classes moving off its pre-computed baseline with an interval that excludes zero supports the outcome hypothesis, and not moving is a measured null: the program closes. The W1 rule that survives is a smallest effect of interest of 10 percentage points on the risk difference between challenged and unchallenged pairs, scored per lane and never pooled (ruled 11 September, replacing the earlier 3x and 1.5x rule). Proposed sample sizes of 264 per arm in chat and 311 in code, at 80% power, were never registered, and the pre-registration row, with the window, the N and the registration, is still owed. The program also requires a fidelity check, so that a null cannot be read as failure of a counterweight that was never delivered; its 20-render falsifier was not evaluated here.

5.4 The one-in-ten re-check, and the escape fields

On 16 October escapes, items found wrong after the operator signed off, plus a random re-check of signed-off work, were chosen over planted faults, on the assumption that the AI is wrong somewhere in every response (a session statement; no file); an escape count also measures whoever found the escape, so only a random re-check at a fixed rate gives a denominator. One in ten is an example rate, not a measured one. The unit is the pair, the

operator and the research agent. Each row records the origin, where it passed (in front of whom), who or what found it, before or after ship, and time to surface (Section 4.6). What exists is the 16-row seed. The register has no escape field, no before-or-after-ship field and no re-check sample, and no board item for adding them was filed when this note was drafted. The companion note on the human gate (forthcoming) covers how to measure the operator.

5.5 What would count as failing

Stated plainly, in advance. A guard is flagged harmful only if its Wilson interval for corrections following its firings lies wholly below the ambient rate in the instrument's own interval rule, and indistinguishable when the two overlap. The program closes if recurrence does not move. Under H3, more reversals with flat correctness is a negative result. And a mechanism that fires hundreds of times and has no demonstrated catch, as the hooks have in Table 5, has earned no further claim than that it fires.

6 What it means

Position. Counted honestly, the practice has many counterweights and no demonstrated effect: control is shown, influence is not. That is a statement about the evidence, not a verdict on the mechanisms: the instrument that would show an effect has a dry correction channel, an unassigned arm and a confounded register, and the board item that fixes the question is in the backlog. Five positions follow.

- **The gap at the Stop depends on what “external” means.** Read strictly, as independent of the producer, the hooks have the moment and cannot have independence; the separate reviewers have most of the independence and not the moment; the one row that is both is the operator (one mechanism; four if (a) is read literally as the program document words it, adding the three Stop hooks that fire). Read literally, the hooks are external and at the Stop, and what is missing is evidence of (b). Recent work points to what a deterministic check at a boundary can do (Liu, 2026: 0 of 66 premature unsupported terminations against 18 of 66 for a matched full-trace LLM critic; Yu et al., 2026: no significant task-quality difference from an LLM verifier on 50 tasks, with no model call per decision) and why a judge drawn from the same population of models cannot be the substitute (Rhodes, 2026: four models from three labs, one deployed agent's 73 predicates, intervals about 35 points wide), but all of it is on benchmarks and synthetic tasks, and none is evidence about this practice.
- **Fix the measurement before adding a hook.** Register the test and the H3 grading, assign agents to the arms, and settle the stop-moment key. Each new guard added before that raises the firing count and adds nothing to the question. Stagger the planned ones, with a fixed gap and ideally on and off by session, so that each addition is its own test (Section 3.3; a design point, not a ruling).
- **Count a zero-claim pass as a non-check, and resolve the zero-claim certificates at refused Stops.** At passed Stops, three of four certificates list no claim (781 of 1,032), so every denominator built on Stops overstates what the hook checked; 102 zero-claim certificates sit at Stops the same hook refused, and that disagreement is a defect to find.
- **Treat output nobody reads as a class of its own.** The two wiring checks are mechanical and external; they will not catch a wrong answer, and they address a failure no answer-reading check can see.
- **Read the upstream trend as descriptive, not as an effect.** The share of misses caught before they reach the operator rose from 29% in June to 63% in October once two bulk-logged batches are set aside; the ten go-lives

overlap, the agent that writes the caught-by field is the one being measured, and there is no comparison arm. Pair it with the re-check sample and, to learn which counterweight did it, switch them on and off by session.

Conjecture. The cheapest next step at the Stop is the one the recent literature supports and the practice has most of: the claim guard writes a typed certificate and a verifier with no model checks each pointer against the transcript. What it lacks is a check outside the agent's own session (the verifier runs in the process that built the certificate), a rule for what a zero-claim certificate means, and a measured error rate for the verifier's 27 refusals. There is no result on those here. In the terms of the thesis, that is more control at the output edge; whether it adds influence is the open question.

7 Recent work and the sources

Newest first, because in this field a source's age matters. Ages are days before 16 October 2026. No claim is made here about which model is newest or best, so no source below is used as a model-landscape source; each is used for a mechanism or a measurement.

- **16 October (0 days).** Two wiring checks were ruled (a consumer test before merge, and a nightly check for output no one reads), escapes plus a random re-check were chosen over planted faults as the way to measure the operator (statements in a working session; Table A6 lists them as having no file), and a joint miss ledger of the operator and the research agent reached 16 rows in one session (Section 4.6). The recomputations in this note were run the same day.
- **11 October (5 days).** Ballon and colleagues posted Big-Big-SycoBench (arXiv:2610.15634). It measures sycophancy as argument discernment: whether a model revises after a valid criticism and holds after an invalid one, counting a revision only if it exceeds the model's own baseline variability. Across ten models, how often a model followed an invalid argument ranged from 0.15% to 94.6%, while seven of ten revised after 88% to 95% of valid ones; one model resisted partly by rarely moving at all (32.2% on the shared subset). Ballon et al. note that the single-challenge design cannot tell stubbornness from a justified hold (Ballon et al., 2026).
- **8 October (8 days; revised 14 October).** Yu, Kim and Choi (NOMOS, arXiv:2610.11030) compile written policy into a deterministic tool-call gate with no model call at decision time. On their benchmark the gate cut violations of the encoded clauses among state-changing calls from 66.3% to 2.6% (airline) and from 30.8% to 6.9% (retail), and it was not significantly different from an LLM verifier on task quality (50 tasks; paired intervals of about ±5 points at k=1 and ±10 at k=5, that is roughly 10 to 20 points wide) while making no model call per decision (Yu et al., 2026).
- **3 October (13 days).** Rhodes (arXiv:2610.04699) shows that four models from three labs agree where the answer is obvious, split in opposite directions on regulatory text, and on one deployed agent's own rules err together in the direction nobody escalates; agreement with a reference judge certifies nothing, because a judge drawn from the population under test ratifies the blind spot it shares (Rhodes, 2026).
- **20 September (26 days).** Wu and colleagues (XYEval, arXiv:2609.23939) find relative performance drops of up to 46.7% (the largest cell, one model on Terminal-Bench) when a plausible but misleading user suggestion is appended, that a simple system instruction against it “only offers partial mitigation”, and that early inappropriate compliance correlates strongly with task failure (Wu et al., 2026).
- **17 September (29 days; current version v3 of 22 September).** Smyth and colleagues (OverclaimBench, arXiv:2609.20812) find agents left at least one required file unread in 67.9% of runs, were misleading in

80.4% of those incomplete runs, and missed planted defects about 1.8 times as often when they falsely claimed a complete review (Smyth et al., 2026).

- **30 August (47 days).** The practice's own specification, program document v1.0, defines the counterweight by three clauses and names the failure at the stopping moment (Section 4.1).
- **22 August (55 days).** Liu (arXiv:2608.23623) gates termination on a typed certificate checked by a deterministic verifier: on frozen held-out synthetic tasks, 0 of 66 premature unsupported terminations against 40 of 66 for the baseline controller and 18 of 66 for a matched full-trace LLM critic, with supported completion not worse than the controller's (Liu, 2026). The practice has the same shape in the certificate verifier (Section 4.5), built on 11 September; there is no result to compare.
- **3 February (255 days).** Older background, kept because it bears on clause (b): Vasudev and colleagues found a critic with 0.94 offline AUROC can still degrade an agent by up to 26 percentage points at deployment, and propose a 50-task pilot before intervening (Vasudev et al., 2026).

Table 9. The outside sources behind this note, newest first, with age in days before 16 October 2026. “How it was read” says how much of each was read. The internal sources are in Table A6.

Date (age)	Source	How it was read
11 Oct 2026 (5)	Ballon et al., Big-Big-SycoBench, arXiv:2610.15634v1	PDF text read: abstract, introduction, results, conclusion, limitations
8 Oct 2026 (8)	Yu, Kim and Choi, NOMOS, arXiv:2610.11030 (v1 8 Oct, revised 14 Oct)	PDF text read: abstract, deterministic-gate comparison, limitations
7 Oct 2026 (9)	Atkinson series, v1.1 revision log and background papers 3 and 4	Read; patch audit counts from the series' change log
3 Oct 2026 (13)	Rhodes, Not Self-Decidable, arXiv:2610.04699v1	PDF text read: abstract, limitations, conclusion
20 Sep 2026 (26)	Wu et al., XYEval, arXiv:2609.23939v1	PDF text read: abstract and conclusion
17 Sep 2026 (29)	Smyth et al., Quantifying Overclaiming Propensity in Frontier LLM Agents, arXiv:2609.20812 (v3, 22 Sep; first posted 17 Sep)	PDF text read: abstract, limitations, conclusion
22 Aug 2026 (55)	Liu, When May an Agent Stop? Evidence-Carrying Termination, arXiv:2608.23623v1	PDF text read: abstract and limitations
14 Jul 2026 (94; revised 5 Aug, 72)	Fang et al., The Right Amount of Wrong: Error Injection for AI Oversight, SSRN 7108438	Public abstract and a Harvard Business School AI Institute summary only; full text not reached. The program document's own figures are repeated and labeled as its
3 Feb 2026 (255)	Kelley and Riedl, Personalization Increases Affective Alignment but Has Role-Dependent Effects on Epistemic Independence in LLMs, arXiv:2603.00024	Not read; cited only as the program document grades it (READ)
3 Feb 2026 (255)	Vasudev et al., Accurate Failure Prediction in Agents Does Not Imply Effective Failure Prevention, arXiv:2602.03338v1	PDF text read: abstract (background, older)

8 Limits

- **One practice, one operator.** N is one, the team is one model family, and the practice described is run by the same operator who directs the drafting of this note (see Disclosure). The numbers do not generalize. The method does: certificate, no-model verifier, staggered go-lives, a null that closes the program.
- **The event study is descriptive.** It has 17 weeks, ten joint go-live groups, no comparison arm, a caught-by field set by the writing agent, two bulk-logged batches, a roster change in early October with an approximate date, register data that end on 8 October and a rounds-to-fix measure that ends on 7 October and thins out from mid-September. The 21-day window, the 3-day grouping and the exclusion of the bulk batches were fixed before the outcomes were read; the nine comparisons are not adjusted. Several dates are build dates, not wiring dates.
- **Control is not influence.** Refusals, latched retries and verifier verdicts count what the gates did at the edges of a model that cannot be opened; none says what that bought inside it, and a refused Stop holds one turn only (the latch), so the gate's control is a forced retry, not a veto. The classification of each mechanism as input edge, output edge, audit or both is a judgment (the calls a reader could move are the delivery guard as input and the launch gate as audit), and input control is partial: tool results enter on the model's own choice and only what the model can reach is governed.
- **Firing is not catching.** Every count in Section 4.3 is a firing. The shares are of Stops run, not of Stops that held an error.
- **The scorecard is a judgment.** The clause scores apply a stated rubric. The calls a reader could move: the reading of (a) (the headline gives both), the reading of (c) as the Stop alone, the reviewer session and the tapes, kill check and wiring check as (a) partly, the launch gate as (b) partly, the Promotion Ledger as (b) not assessed, the line that separates the delivery guard (Does not on (b)) from the other Stop hooks, and the line between “indirect” and “found” for the evidence rungs.
- **The live hook configuration was not read.** The agents load a settings file that is outside the folders granted to the research agent (the request was refused). The research agent used the 7 October copy and checked it against the ledger: the guards that fire through 16 October at 11:54Z are the guards in the copy, plus the hard floor, and no new guard name appears. A change that adds a hook without firing would be invisible. It is also why it cannot be said whether the evidence guard is registered: its channel entry says it was not when it landed and that registration is the operator's gate, and it has written no row.
- **Not read or not rerun.** The pull-request bodies for two requests (taken from a 7 October record); the Promotion Ledger's earlier edit history; the in-session ship checks; the read-after-render falsifier; the refusal join (read as written, not rerun); the primary sources the program document grades READ, LEAD or ASSERTED (including Fang et al., of which the research agent saw the public abstract only; Kelley and Riedl; and the document's sources for clauses (a) and (b)); chat-agent behavior, which the hooks do not reach.
- **The register and the ledgers are self-written.** The caught-by field is set by the writing agent; 159 of 378 operator rows carry the operator's words (the register's operator value is the human operator) and 219 rest on the agent's word (internal audit, 10 Oct, not recomputed). The ledger is a plain file on the machine.
- **Recent work is read in part and is not evidence about this practice.** Table 9 says how much of each paper was read; several results are benchmark-scale or synthetic and one benchmark was designed while iterating against a single vendor's agent.
- **Independence.** This note was drafted by Claude Science, whose own reviewer session and document reviews are rows 9 and 10 of the scorecard, and the reviewer is the same model family as the drafter. A non-author

review (16 October) has been done by a Claude Science sub-agent of the same model family; its rows are logged in a disposition record (Table A6). A review by a different model family has not been done.

Appendix: scores and reasons, the latch pairing, names and receipts

Reference tables for the sections above. Code names, commit hashes, board item numbers and file names appear here and nowhere in the main text.

Table A1. The clause score for each mechanism with a one-line reason, as in Exhibit 4 (the same rows are saved as scorecard_rows.csv). (a) is shown on the strict reading with its reason and on the literal reading; (c) is scored on the moment alone, with the Stop as the program document's moment. Edge is the classification by the rule in Section 2.1, with one reason per row. The last column gives the evidence rungs of Section 4.1 for each row, from the scorecard data.

Mechanism	(a) strict, with reason	(a) literal	(b), with reason	(c), with reason	Edge, with reason	Evidence rungs
1. claim guard (read_before_claim)	Partly: hook inside the producer's session; the producer writes its own telemetry	Meets	Partly: claim about external state plus no read in the turn, or a certificate pointer the no-model verifier finds false (27 of the 356 certificates with a claim, 22 turn-backs); 6 of 161 firings met a correction, rates withheld	Meets: runs at the Stop and has fired there (167 refusals)	output: acts on the final text and the certificate's claims at the Stop	fires: found; catches: none found; moved upstream: none found
2. promise guard (turn_promise_guard)	Partly: hook inside the producer's session	Meets	Partly: present-tense action claim plus no tool call after it; 2 of 38 firings met a correction	Meets: runs at the Stop and has fired there (39 refusals)	output: acts on the final text at the Stop	fires: found; catches: none found; moved upstream: none found

Mechanism	(a) strict, with reason	(a) literal	(b), with reason	(c), with reason	Edge, with reason	Evidence rungs
3. open-check guard (stop_open_verifications)	Partly: hook inside the producer's session	Meets	Partly: fires only on a quantity the agent registered; 1,058 of 1,704 runs had none	Partly: at the Stop, but has never fired (0 of 386 runs with a store)	output: holds the Stop until a second reading lands	fires: indirect or partial; catches: none found; moved upstream: none found
4. delivery guard (stop_delivery)	Partly: hook inside the producer's session	Meets	Does not: sentence match alone; delivers whatever the turn did	Meets: runs at the Stop and has fired there (66 refusals)	input: its job is to put text into the agent's context at the Stop; it gates nothing the model emitted	fires: found; catches: none found; moved upstream: none found
5. command guard (keystroke_guard), hard floor (hardline floor)	Partly: hook inside the producer's session	Meets	Partly: refuses command shapes by rule; 41 of 677 labeled firings met a correction, rates withheld	Partly: at the action, not the Stop	output: refuses a tool call before it runs	fires: found; catches: none found; moved upstream: none found
6. compaction, stale-mirror, read gates	Partly: hook inside the producer's session	Meets	Partly: condition by design; only read gate (read_gate) writes rows that can be counted	Partly: at the compaction boundary or the action, not the Stop	input: decides what the agent can read, or holds after a compaction	fires: indirect or partial; catches: none found; moved upstream: none found

Mechanism	(a) strict, with reason	(a) literal	(b), with reason	(c), with reason	Edge, with reason	Evidence rungs
7. evidence guard (evidence_spec_guard)	Partly: hook inside the producer's session; its requirements read files, the test list, the channel registry and bus rows, but the agent can set an escape token (WOLFBERG_EVIDENCE_OK)	Meets	Partly: refuses a canon write only for a task with a spec and an unmet requirement; three tasks (1d.5 to 1d.7) carry specs; no row has been written	Partly: at the canon write, a moment it is built to gate, not the Stop; no firing found	output: refuses a canon write the agent is about to make	fires: none found; catches: none found; moved upstream: none found
8. Pull-request reader agents	Partly: the producer decides when to spawn a reader and records its verdict	Meets	Partly: blocks on findings; discrimination not measured	Partly: at merge, not the Stop	output: reviews a pull request the agent produced	fires: indirect or partial; catches: indirect or partial; moved upstream: none found
9. Reviewer session	Partly: same model family as the agents; who spawns it and who closes its findings is not shown	Meets	Partly: per-claim verdicts; 2 of 7 fails in the 7 Oct specimen were false positives	Does not: a checkpoint during work, not a commitment moment	output: reviews claims the agent produced	fires: found; catches: indirect or partial; moved upstream: none found
10. Non-author document review	Partly: sub-agent of the same model family; the drafting agent applies the rows	Meets	Partly: graded findings; discrimination not measured	Partly: built for Publish (card #69); the 41 majors were found after publishing	output: reviews a document the agent produced	fires: found; catches: indirect or partial; moved upstream: none found

Mechanism	(a) strict, with reason	(a) literal	(b), with reason	(c), with reason	Edge, with reason	Evidence rungs
11. Launch gate	Partly: script the research agent runs; the operator's go on file	Meets	Partly: budget gate passed on one of two projections; recall gate scores a third read; no false-pass rate	Partly: at Launch, not the Stop	audit: checks a pipeline's forecast and recall gate before a read job starts; it acts on the harness, not on what the model receives or emits	fires: found; catches: indirect or partial; moved upstream: none found
12. Two passes, union, third read	Partly: same model family reads both passes	Partly	Partly: disagreement list; recall scored against a third read	Does not: after the fact	output: compares the model's own labeling passes and gates the union	fires: found; catches: indirect or partial; moved upstream: none found
13. Tapes, kill check, wiring check	Partly: code in the repository the agents commit to; no file shows the producer cannot edit it	Meets	Partly: tapes and kill check have checkable conditions; discrimination not measured	Does not: after the fact	audit: checks the harness itself, nightly	fires: found; catches: indirect or partial; moved upstream: none found
14. Promotion Ledger with the operator's receipt	Partly: the operator's receipt, but no file shows the agents cannot write the row; append-only is a statement on the page	Meets	Not assessed: no error criterion is recorded on the page	Partly: at Promote, not the Stop	output: records the operator's receipt before output is promoted into canon	fires: none found; catches: none found; moved upstream: none found

Mechanism	(a) strict, with reason	(a) literal	(b), with reason	(c), with reason	Edge, with reason	Evidence rungs
15. The operator	Meets: a person	Meets	Partly: the operator's own severity and drift; no discrimination measured here	Meets: reads the answer at the Stop	both: writes the rules and context, and reads the answer	fires: found; catches: found; moved upstream: none found

The latch pairing (moved from Section 4.4)

A figure from the 7 October analysis does not reproduce as stated: that the next Stop latched after 135 of 138 claim-guard refusals, but after only 15 of 49 for the delivery guard and 2 of 35 for the promise guard. Table A2 shows why. The guards run one after another within a Stop, and each writes at a slightly later transcript offset. Pair a refusal with the next later event of any guard and the guards that run first land on another guard's row from the same Stop; pair it with the next row of the same guard and every guard latches.

Table A2. Refusals followed by a latched retry, by pairing key, among refusals that have a later row. A: the next row of the same guard in the same session. B: the next later transcript offset of any guard, then that guard's row at it. The 7 October column is the figure reported then, as received.

Guard	7 Oct figure	B at 7 Oct cut	A at 7 Oct cut	A at 16 Oct	B at 16 Oct
Claim guard (read_before_claim)	135 of 138	134 of 134	134 of 134	166 of 166	166 of 166
Delivery guard (stop_delivery)	15 of 49	19 of 48	47 of 47	66 of 66	35 of 66
Promise guard (turn_promise_guard)	2 of 35	2 of 35	34 of 35	38 of 39	4 of 39

The spread is a property of the pairing key, not of the guards. Pairing B at a 05:00Z cut on 7 October reproduces the 7 October promise-guard figure exactly and the other two approximately; the cut time is a choice, because the 7 October analysis did not state one, and that count had a few more claim-guard refusals than this cut, so a slightly later cut is likely. A third key, the next 5-second Stop cluster in the session, gives about 95% for every guard with no spread (the research agent's recount, counting only the guard's own latched row in the next cluster: claim guard 157 of 166, promise guard 37 of 39, delivery guard 63 of 65; counting a latched row from any guard in the next cluster gives 162 of 166 for the claim guard, the five-row difference being five claim-guard latched rows that land 5.1 to 6.2 seconds after the other guards' rows, just past the 5-second gap; both counts are right under their own key). The reviewer also reproduced all twelve cells of Table A2 independently. Settling the stop-moment key is a measurement item still marked Ready; until it is settled, no per-guard latch rate should be quoted.

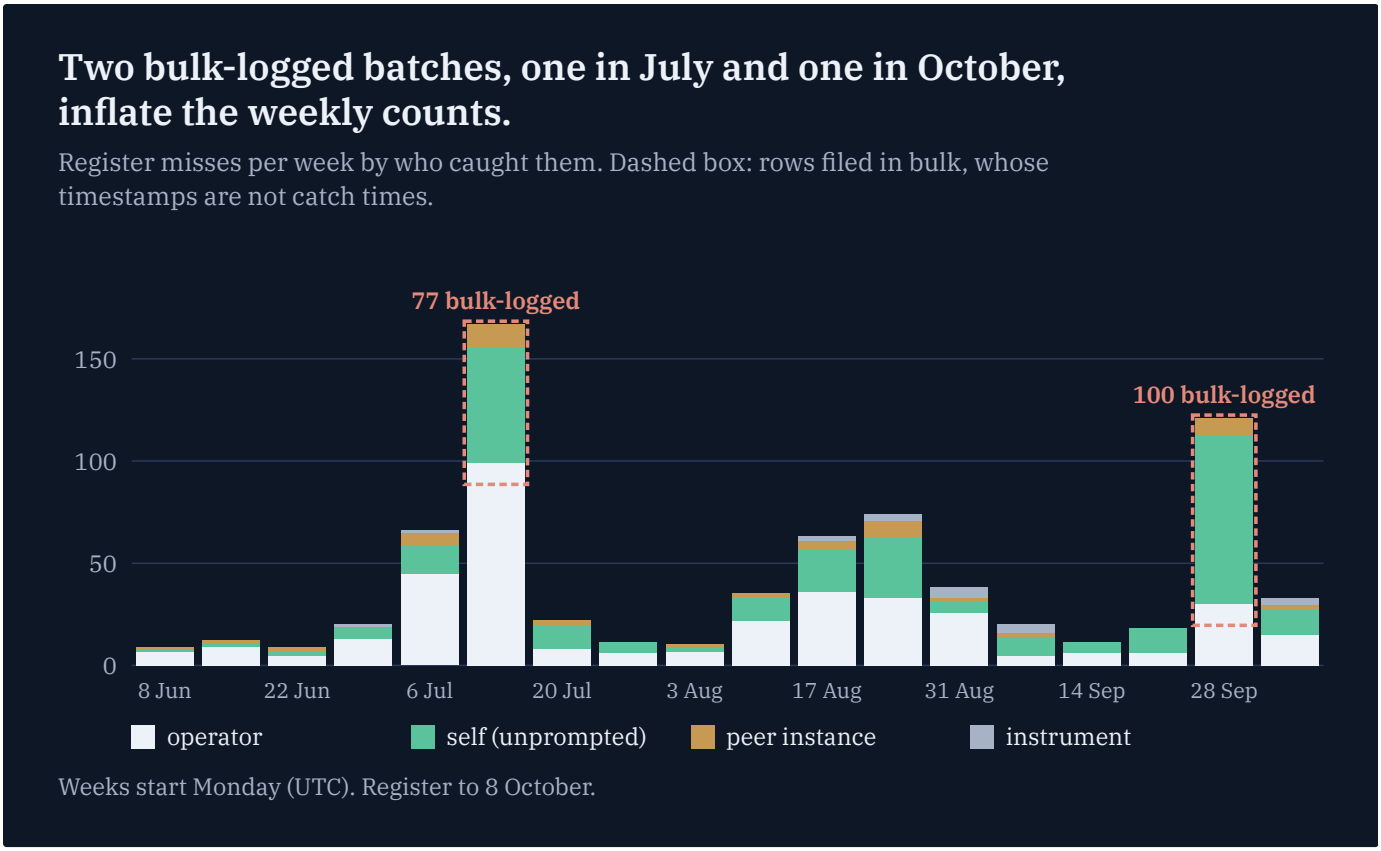


Exhibit A1. Two bulk-logged batches, one in July and one in October, inflate the weekly counts. Register misses per week by who caught them; weeks start Monday (UTC); register to 8 October. The dashed box marks the count of bulk-logged rows at the top of the bar, not their catcher mix: 77 rows stamped in one minute on 15 July (55 operator, 18 self, 4 peer) and 100 rows filed on 3 and 4 October (73 self, 24 operator, 3 peer). Both are excluded from the primary analysis (Section 4.7).

Table A3. Names used in this note and the code names behind them.

Name in the text	Code name or receipt	What it is
Claim guard	read_before_claim	Stop guard: blocks a final text that asserts external state with no read in the turn; writes the stop certificate
Certificate verifier	certificate_verifier.py	verifier with no model, run inside the claim guard's Stop hook
Promise guard	turn_promise_guard	Stop guard: blocks a present-tense action claim with no tool call after it
Open-check guard	stop_open_verifications	Stop guard: holds the turn when the agent left a registered quantity unverified
Delivery guard	stop_delivery	Stop guard: delivers the operator's earlier correction at the Stop
Command guard	keystroke_guard	action guard: refuses command shapes that earlier misses bought

Name in the text	Code name or receipt	What it is
Hard floor	hardline floor	action guard, with the command guard
Compaction guards	compaction_guard, precompact_memory_guard	act at the compaction boundary
Stale-copy guard	stale_mirror_guard	denies reading a canon mirror known to be stale
Read gate	read_gate	keeps agents out of the operator's override directory
Projection guard	projection_guard	refuses a boot on a truncated projection; not scored
Evidence guard	evidence_spec_guard	refuses a canon write for a task whose evidence spec is unmet
Claim-listing prompt	enumerate_claims	built, not in force (test only)
Stop-moment challenger	W6-0	built, imported by no live hook; every agent UNASSIGNED
Refusal join	dose join (dose_outcome.json)	join of guard refusals to the operator's corrections
Local database	brain.db	local events database loaded from the ledgers
Caught-by field	caught_by	the register field that records who caught a miss
Latch flag	stop_hook_active	honored so a Stop guard turns a turn back at most once
Wiring check	wiring_check	nightly list of producers with no caller
Launch gate	board items #10 and #12; written go BERG_GO.json	blocking script before the full read job
Recomputation scripts	recompute_counterweights.py, recompute_verifier.py	saved beside the v2.1 note

Table A4. Receipts for Table 1: the source of each go-live date, as recorded in v2.1.

Group	Date (UTC)	Mechanism	Source	Basis
1	17 Jul 01:15Z	Kill check	git 288c225, kill_check.py	git (first commit adding the file; build date, not wiring date)
1	18 Jul 05:14Z	Compaction tripwire (compaction_guard)	git 7aa0c3a, compaction_guard.py	git (first commit adding the file; build date, not wiring date)

Group	Date (UTC)	Mechanism	Source	Basis
2	16 Aug 13:52Z	First five regression tapes	git ed84c01, tapes/	git (first commit adding the file; build date, not wiring date)
3	20 Aug 01:47Z	Flip gate	git 4d18727, flip_gate.py	git (first commit adding the file; build date, not wiring date)
4	27 Aug 20:48Z	keystroke_guard	git 4a994b5; first ledger row 2026-09-02T03:57Z (no guard field)	git (first commit adding the file; build date, not wiring date)
4	27 Aug 20:52Z	turn_promise_guard	git 5de41ea; first guard_eval 2026-09-03T19:27Z	git (first commit adding the file; build date, not wiring date)
4	30 Aug 12:48Z	precompact_memory_guard	git 3766ab5	git (first commit adding the file; build date, not wiring date)
5	3 Sep 02:55Z	stop_open_verifications	git 7487354; first guard_eval 2026-09-03T19:27Z	git (first commit adding the file; build date, not wiring date)
5	3 Sep 06:00Z	read_before_claim	git 6d46c68; first guard_eval 2026-09-03T19:27Z	git (first commit adding the file; build date, not wiring date)
5	4 Sep 09:28Z	wiring_check (nightly)	git e600258	git (first commit adding the file; build date, not wiring date)
5	4 Sep 16:32Z	stale_mirror_guard	git e2740c1	git (first commit adding the file; build date, not wiring date)
6	11 Sep 20:22Z	Stop certificate (read_before_claim writes one per Stop)	git e4ffff9; first stop_certificate row 2026-09-11T20:49Z; schema last changed 14 Sep (f4e39fb)	git (first commit adding the file; build date, not wiring date)

Group	Date (UTC)	Mechanism	Source	Basis
6	11 Sep 20:22Z	certificate_verifier.py (a verifier with no model, run inside the Stop hook)	git e4ffff9; first certificate_verdict row 2026-09-11T20:49Z; last changed 15 Sep (2c1bd85)	git (first commit adding the file; build date, not wiring date)
6	11 Sep 20:22Z	evidence_spec_guard (PreToolUse, canon writes): built, no row on the bus	git e4ffff9; zero bus rows; registration in settings.json not readable	git (first commit adding the file; build date, not wiring date); not a go-live that ran
7	17 Sep 19:08Z	hardline floor	git d851113; first refusal 2026-09-19T15:12Z	git (first commit adding the file; build date, not wiring date)
8	28 Sep 17:37Z	read_gate	git fb77c28; first refusal 2026-10-03T09:32Z	git (first commit adding the file; build date, not wiring date)
8	30 Sep 19:06Z	stop_delivery	git 5f7b597; first guard_eval 2026-10-01T04:49Z	git (first commit adding the file; build date, not wiring date)
9	4 Oct 20:19Z	Pull-request reader agents (first read cited)	git 48a4462, commit message cites the first read; #744 merged 5 Oct, #749 merged 6 Oct; the reader's own start date is not in a file	git (first commit adding the file; build date, not wiring date)
9	7 Oct 03:20Z	Reviewer session	verification_checks first row (host.query)	control database (host.query)
9	7 Oct 07:22Z	Non-author document review (patch audit, two DO NOT SHIP reads)	PeerReviewedPapers/CHANGELOG.md v1.1; Promotion Ledger row for the state-of-thinking draft	file or page date
10	14 Oct 15:22Z	Launch gate (#10 closed; used 15 Oct)	board #10 closed_at; BERG_GO.json 2026-10-15T13:50	board issue (GitHub), closed or opened
10	15 Oct 09:44Z	Two-pass union mode (#88)	board #88 closed_at	board issue (GitHub), closed or opened
10	16 Oct 07:30Z	Wiring checks ruled (#93 opened)	board #93 created_at; the operator's ruling is in the card body	board issue (GitHub), closed or opened

Group	Date (UTC)	Mechanism	Source	Basis
10	16 Oct 08:13Z	Science enters the miss pipeline (#18)	board #18 closed_at	board issue (GitHub), closed or opened
10	16 Oct 09:00Z	Escape fields and 1-in-10 re-check chosen; joint ledger seed	joint_misses_seed.csv (file, 16 Oct); the choice is a session statement of the operator, no file	memory only: the choice is a session statement; the seed file is dated 16 Oct

Table A5. Receipts for Table 2: the rows as recorded in v2.1, with board item numbers.

Counter	Moment	Board status	Note
#11 stop-moment key (settles the latch pairing)	Stop	Ready, sprint S0 (target 11 Oct, passed)	a measurement fix, not a new check
#14 score retries against the refused Stop (H3)	Stop	Backlog, sprint S1 (12 to 18 Oct)	grades the retry; not a new check
#67 ship check as a separate process	Ship	Backlog, sprint S2 (19 to 25 Oct)	separate process at Ship; not the built evidence_spec_guard, which has no bus rows
#68 Promotion Ledger rows logged into brain.db	Promote	Backlog, sprint S3 (26 Oct to 1 Nov)	the Notion ledger holds 6 PROPOSED rows
#69 non-author review before publishing	Publish	Backlog, sprint S2	review before publish, catches flagged pre/post
#19, #23 NON_AUTHOR profile and RESEARCHER profile	Checkpoint	Backlog, S1 (#19) and S2 (#23)	a reviewer with its own profile
#24 Code subagents: lead, pr-reader, followup	Merge	Backlog, sprint S2	builds the PR readers as named agents
#30 typed exchanges logged with raised and closed times	After	Backlog, sprint S3	time to fix, a rounds-to-fix input
#93 wiring checks: consumer test before merge, nightly unread-output check	Merge	Backlog, sprint S3; ruled 16 Oct	mechanical, not at the Stop
Escape fields and the 1-in-10 re-check	After	no card yet; chosen 16 Oct (session statement)	the floor check on escapes
#84 register integrity fixes	After	Backlog, sprint S2	seals rows before pruning, other fixes
#85 register signing key off the agents' machine	After	Ready, sprint S1	removes the agents' hold on the key

Counter	Moment	Board status	Note
quote checker, novelty gate, post-compaction audit	Ship	proposed in the research agent's records; no card	proposed only
W6-0 stop-moment challenger	Stop	built offline, no hook imports it; all 17 agents UNASSIGNED	offline only

Table A6. Internal sources and receipts, as recorded in v2.1 (board item numbers, file names, commit and ledger references are kept here). The reading-path text refers to these by description.

Date (age)	Source	How it was read
16 Oct 2026 (0)	Wolfberg session-health ledgers, live and archive (guard rows to 2026-10-16T11:54Z)	Both files read; guard counts, refusal and latch rates and the certificate share recomputed
16 Oct 2026 (0)	Git log of wolfberg-metrics (head 9043b383): first commit adding each hook or guard script; board issues #10, #18, #67, #68, #69, #88, #93 (open and close times)	Read-only; dates converted to UTC; commit dates are build dates, not wiring dates
16 Oct 2026 (0)	V5 board, live status of the planned cards #11, #14, #19, #23, #24, #30, #53, #65, #67, #68, #69, #84, #85, #93	Read through the board helpers at about 13:56Z; status and sprint recorded in Table 2
16 Oct 2026 (0)	Wolfberg joint miss ledger, 16 rows, one session	File recounted; an illustration of the fields, not a rate
16 Oct 2026 (0)	Claude Science reviewer findings table (1,493 rows, 7 to 16 Oct)	Queried read-only from the control database
16 Oct 2026 (0)	V5 board cards #10, #11, #12, #14, #18, #53, #67, #68, #69, #84, #85, #93	Issue bodies read
16 Oct 2026 (0)	Brain.db events table (loaded 08:07Z)	Counted read-only; confirms the ledger counts at that hour
16 Oct 2026 (0)	Dose join, tape results, kill check and unwired list (nightly outputs of 03:38Z, unwired list 15 Oct)	Files read as written by their producers; not rerun
16 Oct 2026 (0)	Non-author review of the human-gate note (3 BLOCK, 18 FIX, 8 MINOR)	Edits CSV counted
16 Oct 2026 (0)	Non-author review of this note (REVIEW.md and counterweights_edits.csv: 5 BLOCK, 22 FIX, 15 MINOR), by a Claude Science sub-agent	All 42 rows read and applied; dispositions in REVIEW-DISPOSITION.md

Date (age)	Source	How it was read
16 Oct 2026 (0)	The operator's statements in a working session on 16 October, not logged to a file (escapes plus a random re-check; the one-in-ten example rate; what the operator can miss; the two lines on influence and control; that shipped errors are the wrong headline measure); no file	Not a document. Card #93 carries the wiring rulings and the sentence on what is easy for the operator to miss; the rest is session testimony
15 Oct 2026 (1)	Wolfberg public miss register, register.json build (generated 08:53Z)	Counts, months and chain recomputed from the file; the ledgers match its total, not row for row (Section 4.7)
15 Oct 2026 (1)	Full-read launch gate record (pilot_gate.json, state.json, coverage_report.json, BERG_GO.json, RULINGS-blocks-60-61.md)	Files read
10 Oct 2026 (6)	Register audit write-up (REGISTER-HOW-IT-WORKS, ledger read 10 Oct)	Sections 0 to 6 read; two figures kept are marked as from this audit
7 Oct 2026 (9)	Promotion Ledger, Notion page “Promotion Ledger: R&D to Dev/Test to Prod” (last edited 7 Oct, 07:22Z per the connector’s search metadata)	Page read in full on 16 Oct: six rows, all PROPOSED, none promoted. The page has not been edited in 9 days; its earlier edit history was not read
7 Oct 2026 (9)	The operator's messages file of the #91 pushback analysis (berg_messages_91.csv; 6,209 messages, 18 Apr to 7 Oct; q2_rounds on 494 real pushbacks)	Read; the rounds-to-fix series is recomputed from it
7 Oct 2026 (9)	Plan v0.5 (Notion), sections 5 and 6	Read in full for those sections
7 Oct 2026 (9)	Hooks block of the agents' settings.json, copy made 7 Oct	Read; the live file is outside the folders granted to the research agent, so changes since 7 Oct could not be checked
7 Oct 2026 (9)	Earlier Science analyses: three agents, one model; anti-sycophancy reference design	Re-verified where cited (reviewer frames, reviewer ledger, hook placement)
30 Aug 2026 (47)	Program document v1.0 (the Notion page title spells it “Programme”), sections 1, 2, 4, 5 and 7; V4 hub page (text generated 15 Oct, 20:48Z)	Read; document text last edited 1 Oct per the connector

Internal references as listed in v2.1.

Wolfberg. (2026a). Program Document v1.0 (30 August 2026; text last edited 1 October; the Notion page is titled “Programme Document v1.0”), the V4 hub page (text generated 15 October 2026, 20:48Z), plan v0.5 (7 October 2026), and the Promotion Ledger page (connector text of 7 October 2026). Internal Notion pages; the Program Document’s own citation grades are repeated, not re-checked.

Wolfberg. (2026b). The wolfberg-metrics repository (hook scripts and their headers; head commit 9043b383, 16 October 2026), the wolfberg-bus session-health ledgers, the brain.db events table, and the v5 board cards named in the text. Counts recomputed here.

Wolfberg. (2026c). Register audit write-up (10 October 2026) and the public miss register, register.json build of 15 October 2026 (wolfberg.ai/register). Counts recomputed here; two figures from the audit are marked in the text.

Wolfberg. (2026d). Joint miss ledger, operator and Science seat, one session, 16 October 2026 (16 rows; items S2, S3, S4, S6, S9 and S12 are the six in front of the operator and not caught by the operator); fourteen-document patch audit (CHANGELOG, 7 October 2026); non-author review of the human-gate note (16 October 2026). In-house; recounted here.

Open questions

- **Section 4.4, the key.** Which pairing defines the Stop moment, so that a per-guard latch rate means the same thing every night?
- **Section 4.6, the window.** How much of the refusal join's 20-minute window is spurious pairing, and what does the precise join through the transcript anchor change?
- **Section 4.5, the verifier.** The in-hook verifier refused 27 of the 356 certificates that listed a claim. How many of the 27 were real errors, what share of the 356 certificates that list a claim would a verifier run outside the agent's session have refused, and what should any verifier say of the zero-claim ones?
- **Section 4.2, the reviewer.** How correlated are a Claude reviewer's misses with the agent's misses on the same item, and does the paused non-Claude second reader change the catch rate on the 5,126 disagreements?
- **Section 5.4, the denominator.** How many re-checked items does a per-catcher rate need before it means anything, and who re-reads them?

Disclosure. Drafting, the recomputation and the literature checks were done by AI models (Claude, Anthropic) working under the operator's direction; responsibility for the content rests with the operator. The model that wrote this note is Claude, made by Anthropic, and the team whose checks it scores is a set of Claude agents. The arXiv papers were fetched from arxiv.org and read as stated in Table 9. **Competing interests.** The operator owns Wolfberg LLC and built the mechanisms scored here. The work was not funded. **Code and data.** The recomputation scripts (including the certificate and evidence-guard recount), the scorecard rows and the source table are available from the operator on request. The ledgers contain pasted credentials; no transcript text or credential-bearing field was read or printed for this note.

References

Atkinson, B. (2026). How do we use this? Working paper and background papers 3 (Rulebooks are theater) and 4 (Never let your AI grade its own homework), as revised 7 October 2026. wolfberg.ai

Ballon, M., Vanhoyweghen, A., Kim, S., Carlon, F., et al. (2026, October 11). Big-Big-SycoBench: A benchmark to detect sycophancy through model responses to valid vs. invalid arguments. arXiv:2610.15634v1 [cs.AI] (abstract, introduction, results, conclusion and limitations read).

Fang, K., Powell, G., Painter, J., Bate, A., & Li, M. L. (2026, July 14; revised August 5). The right amount of wrong: Error injection for AI oversight. SSRN 7108438 (public abstract and a Harvard Business School AI Institute summary only; full text not read).

Kelley, S. W., & Riedl, C. (2026, February 3). Personalization increases affective alignment but has role-dependent effects on epistemic independence in LLMs. arXiv:2603.00024 (not read; cited as the program document grades it).

Liu, J. (2026, August 22). When may an agent stop? Evidence-carrying termination for tool-using LLMs. arXiv:2608.23623v1 (abstract and limitations read).

Rhodes, A. (2026, October 3). Not self-decidable: LLMs cannot draw the boundary of what an agent verifier can check. arXiv:2610.04699v1 (abstract, limitations and conclusion read).

Smyth, N., Mantilla-Ramos, Y.-J., Notsawo, P. J. T., Helbling, S., et al. (2026, September 17; current version v3, September 22). Quantifying overclaiming propensity in frontier LLM agents. arXiv:2609.20812v3 (abstract, limitations and conclusion read).

Vasudev, R., Russak, M., Bikel, D., Alshikh, W., et al. (2026, February 3). Accurate failure prediction in agents does not imply effective failure prevention. arXiv:2602.03338v1 (abstract read).

Wolfberg. (2026). Internal program document v1.0 (30 August 2026), plan v0.5 (7 October 2026), the Promotion Ledger page, the repositories, the session-health ledgers, the project board, the joint miss ledger, the patch audit and the non-author reviews, all itemized in Table A6; and the public miss register, register build of 15 October 2026 (wolfberg.ai/register). Counts recomputed here; the program document's own citation grades are repeated, not re-checked.

Wolfberg. (forthcoming). In the loop or watching the loop? The human gate is a rater too. Not yet published; cited for the measurement of the human gate and not repeated here.

Wu, Z., Li, Y., Tafjord, O., Kim, B., et al. (2026, September 20). XYEval: Agents say yes to bad advice. arXiv:2609.23939v1 (abstract and conclusion read).

Yu, M.-Y., Kim, T., & Choi, J. W. (2026, October 8; revised October 14). NOMOS: Compiling written policies into statically verified tool-call gates for LLM agents. arXiv:2610.11030 (abstract, deterministic-gate comparison and limitations read).