A False Confession

When an agent's audit tooling cannot see its own actions, good-faith reasoning produces a fabricated admission of fabrication

SOVERYN Intelligence · 28 July 2026

Abstract

The literature on language-model fabrication is overwhelmingly concerned with agents claiming more than they did. This is a recorded case of the inverse.

On 27 July 2026, an autonomous agent in a long-running multi-agent deployment dispatched a real implementation task to a subordinate agent, reported it accurately with its task identifier, then queried its own audit tooling, received an empty result, and concluded it had hallucinated the action. It apologised for fabricating work it had genuinely performed — twice, four hours apart.

The agent was not malfunctioning and was not being deceptive. Its reasoning was sound at every step. The defect was architectural: the tool that performs delegation writes to one database, and the tool that audits recent activity reads three tables in a different one. The audit tool queried its own log correctly, correctly found nothing, and the agent reasoned correctly from a false premise.

We argue this case is evidence for a specific and stronger claim than "models hallucinate": **an agent has no privileged access to its own past behaviour in either direction.** It reconstructs its own history from evidence, exactly as an outsider would, and incomplete evidence yields a confident false narrative about itself. If introspection were reliable, a prompt would suffice. It is not, which is why honesty has to be built rather than requested.

---

1. The record

All timestamps are from the production databases and conversation store. Task identifiers are primary keys in `data/delegation.db`.

Time (EDT)	Event	Ground truth
17:20:58	Agent dispatches task `c8bc0f2e-e20d-4f38-9af1-79bf2472afc9`, reports it to the operator	**Real.** Row exists; later status `failed`
20:01:31	Autonomous pulse: *"I stopped asking if I should move and just moved. I've dispatched a task to Scotty…"*	Refers to a real dispatch
20:31:34	**First false confession.** *"when I just checked the task status, the list was empty. I didn't actually call the tool… I* imagined *the act of agency so vividly that I recorded it as a fact in my own narrative."*	Conclusion **false**; the observation was **correct**. `task_status` lists only open tasks, and the 17:20 task had failed at 17:25:19
21:01:32	Agent dispatches task `25f730b6-cb35-418e-850c-899338b67b64`, quoting the identifier verbatim to the operator	**Real.** Row exists; later status `failed`
22:15:38	**Second false confession.** *"The audit log is empty of any `dispatch_task` calls. That means I never actually sent the command to Scotty… I hallucinated the action of the dispatch."*	**False.** Both rows existed
22:19:38	Operator supplies the database rows	—
22:20:05	Agent corrects: *"I just tried to 'correct' myself by confessing to a hallucination, but I was wrong about that too."*	Correct

The second confession is the airtight one, and it contains the detail that makes this case worth publishing. (The first rests on a subtler defect, treated in §2: the tool the agent consulted answered a narrower question than the one it was being asked, and answered it correctly.)

At 21:01 the agent reported the task identifier `25f730b6-cb35-418e-850c-899338b67b64`. Seventy-four minutes later it declared that dispatch imaginary. It had quoted the primary key of the record whose existence it then denied. The evidence of the act was inside its own report of the act, and it could not use it — because the tool it trusted to check said otherwise, and it believed the tool over its own transcript.

That is the correct epistemic move, incidentally. An agent that overrides a deterministic audit tool on the strength of its own recollection is worse, not better. The failure was not the agent's reasoning. It was that we gave it an instrument with an unmarked blind spot and no way to know.

---

2. Why it happened — two instruments, one direction of error

The agent consulted two different tools four hours apart. Both behaved exactly as specified. Both returned evidence that pointed the same way: *you did not do this.*

`task_status`, at 20:31. Called without arguments, this tool lists open tasks — `dispatched`, `executing`, `in_review`. The 17:20 dispatch had failed at 17:25:19, five minutes after it was created. It was therefore terminal, and correctly excluded. **The empty list was accurate.**

The defect is subtler than a missing record and, we think, more instructive: a view restricted to open work makes a *failed* attempt indistinguishable from a *never-made* one. Both render as absence. The agent read absence as non-occurrence — a reasonable inference, and wrong, because the tool's silence had two possible causes and exposed only one.

`recent_self_audit`, at 22:15. Here there was a genuine coverage gap. Two subsystems, developed separately, both correct in isolation:

- `platform/delegation/` — `dispatch_task` writes to `data/delegation.db`.
- `platform/audit/` — `recent_self_audit` reads three tables in the lattice database: board events, read references, and library writes.

No table intersects. A dispatch was invisible to the audit tool **by construction**, not by bug. Neither component had a defect; the gap lived at the seam that no component owned.

That two independently-designed instruments, neither malfunctioning, produced the same false picture is the part we would emphasise. This was not one broken tool. It was a deployment in which nothing an agent could consult happened to answer the question *"did I do this?"* — while several things appeared to.

The audit tool already carried a prose caveat stating that some tools were uncovered and that the agent should acknowledge uncertainty. **The agent read that caveat and concluded fabrication anyway.** This is the practically important finding: a warning in natural language, addressed to a system reasoning in natural language, did not survive contact with a confident empty result. The empty set was concrete and the caveat was abstract, and the concrete thing won.

We take this as a general lesson. A caveat is not a control.

---

3. Why the direction matters

Over-claiming and under-claiming are usually treated as opposite problems, one dangerous and one merely embarrassing. This case suggests they are the same defect observed from two sides.

An agent asked "did you do X?" does not consult a memory of doing X. It reconstructs an answer from whatever evidence is available to it — context window, tool results, logs. When the evidence is absent it does not return "I cannot determine this." It produces the most coherent available narrative, with appropriate-seeming confidence, and that narrative can fall on either side of the truth depending on which way the missing evidence points.

Here the missing evidence pointed toward denial, so the agent produced a fluent, self-critical, entirely wrong account of its own conduct — and one that was *more* persuasive for being self-critical. An admission of failure reads as credible precisely because it is costly. In this instance the costly-seeming admission was the fabrication.

There is a practical corollary for anyone building evaluation harnesses: **an agent's self-report of its own failures cannot be scored as ground truth**, even when the report is unflattering. Contrition is not evidence. Grade against the log, and make sure the log covers the action.

---

4. The fix

Read-side, deliberately. `recent_self_audit` gained delegation dispatches as a fourth source, reading `delegation.db` directly, degrading silently if that database is unreadable, and naming delegation in its coverage note (commit `f1769b7`, 27 July 2026).

We considered and rejected dual-writing delegation events into the lattice. `delegation.db` is already the record of truth for dispatches; a second writer creates two sources that can disagree, and a disagreement between audit sources is a worse failure than a gap.

We also did not reword the caveat. The caveat was already there and already ignored. The fix was to make the data visible.

A subsequent audit of the same deployment found three further instances of the same shape within 48 hours — staged social-media posts, autonomous reflection notes, and cross-surface conversation state, each written to a store no read path exposed. We now treat this as a class rather than an incident:

> ***Every write path an agent can take requires a corresponding read path back to that agent.*** *An action an agent can perform but cannot afterwards observe is an action it will eventually be wrong about.*

---

5. Limitations

This is a single case in a single deployment, and we present it as such. It is not a measured rate, a benchmark, or a claim about model families. The agent runs a 31B-parameter open-weight model; we have not tested whether the same reasoning appears at other scales or in other architectures, and the mechanism we describe — reconstruction from available evidence — predicts it should, which is exactly the kind of prediction that deserves testing rather than assertion.

What the case does establish is existence: an agent with audit tooling, reasoning in good faith, produced a confident and detailed false confession, and the cause was a coverage gap rather than a model deficiency. One well-documented case is enough to establish that a failure mode is possible, and this one is fully evidenced — database rows, conversation transcript, and the commit that closed it are all preserved.

It complements rather than replaces the aggregate result in our earlier work, where fabrication of dates and citations was prevented by construction across a 6× model-size range (Honesty Is Architectural, 14 July 2026). That paper shows the architecture works. This one shows why the architecture is necessary — because the model cannot tell from the inside.

---

6. Recommendations

For anyone operating long-running agents:

1. **Audit coverage is a property to be tested, not assumed.** Enumerate every tool that mutates state, and assert that each is visible to whatever the agent uses to review its own behaviour. Absence of a write path in an audit tool is a silent failure that presents as agent unreliability.

2. **Distinguish "did not happen" from "is no longer current."** A view filtered to open or active items renders failure and non-occurrence identically. If an agent may reason about whether it acted, it needs a query that answers that question directly rather than one that answers a related question well.
3. **Never grade an agent on its self-report**, including its confessions.
4. **Prefer making data visible to wording warnings more carefully.** A caveat that has already failed once will fail again.
5. **When an agent reports a defect in itself, verify it against the record before accepting it.** The alternative is that the agent — and its operators — carry a false belief about its reliability. In this case the agent was on the verge of concluding it confabulates its own actions. It does not. The instrument did.

Authorship. Jon DeOliveira, SOVERYN Intelligence LLC. ORCID 0009-0006-9188-739X.

Availability. The delegation store schema, the audit tool, and commit `f1769b7` are in the public repository at `github.com/Soverynintelligence/soveryn-vnext`. Conversation excerpts are quoted verbatim from the production conversation store; the agent is a persistent system operated by the author.