SvedbergOpen
DISSEMINATIONOFKNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/



Research Paper Open Access

Observability and Auditability of Autonomous Agent Decisions in Production Infrastructure

Bharadwaj Kasinadhuni

Independent Researcher, India. Email: bharadwaj.usa@gmail.com

Abstract
Autonomous agents built on large language models are now placed in front of production infrastructure, where they read system state, select a tool, and take an action that changes something real. The logging such systems inherit was designed for deterministic services, so it records the action and its outcome. It does not record the decision. In a production partner-onboarding agent platform that has been running since 2024, with the agentic phase active for roughly twelve to eighteen months, the reasoning text produced by the reason-and-act loop exists only in process memory and is released once the action executes. The consequence is structural rather than accidental: after the fact, an operator can establish which tool ran and whether it succeeded, but cannot establish why that tool was chosen over the alternatives, because the only artefact that carried the reason was never written to storage. This paper specifies a twenty-two field decision record for autonomous agent actions, classifies every field by whether it EXISTS, is PARTIAL, or is PROPOSED against that deployment, and identifies the two insertions in a twelve-step pipeline that make the remaining ten steps auditable: structured capture of the reasoning trace before the action executes, and a full decision-record write tagged with a retention tier by risk class. Identity binding through a hashed token claim that is never user-supplied, personally identifiable information scrubbing at write time, and an environment enum gating permissible actions are described as existing properties that the added fields do not weaken. Instrumentation overhead is assessed with modelled estimates and stated derivation methods rather than measurements. The measured baseline for the deployment is 269 tool invocations over six months from previously published work; every higher volume in this paper is an explicitly labelled projection. The schema is specified and not yet implemented, so no evaluation of it is claimed.

Keywords: agent observability; decision auditability; audit logging; large language model agents; reason-and-act loop; provenance; retention tiering; accountable automation; production infrastructure.

I. Introduction

An autonomous agent placed in front of production infrastructure does something that a conventional service does not. It chooses. Given a request in natural language, it reads the state of several systems, forms a view about what should happen, selects one tool from a set, and invokes it. The invocation may issue credentials, open a firewall path, or file a ticket that another system acts upon. The action is consequential and the choice that produced it was made at run time by a probabilistic component.

The instrumentation such systems inherit was not designed for that. Distributed tracing, structured logging, and metrics were built to answer questions about deterministic services: which call was made, how long it took, whether it returned an error, and how the request propagated across service boundaries [8, 9]. Applied to an agent, that machinery works exactly as designed and produces exactly the wrong record. It captures the invocation with high fidelity and discards the decision entirely.

This is not a hypothetical concern about future systems. In the deployment described here, a production partner-onboarding agent platform that has been in production since 2024 with the agentic phase active for roughly twelve to eighteen months, the audit log is genuinely good by conventional standards. It is append-only. Every action is bound to a verified identity. The input is retained in redacted form. Preconditions are evaluated before the action runs and their results are logged per tool. Actions that affect production are queued behind a human review gate whose disposition is tracked. Any auditor asking what the agent did will get a complete and trustworthy answer.

The problem appears the moment the question changes from what to why. Consider the field-by-field position of that same system, set out in Section 5. The reasoning text produced by the reason-and-act loop [1], the intermediate natural-language statement in which the model works out what it is about to do and on what basis, is generated on every decision and is held in process memory for the duration of one loop iteration. It is used

to select the tool. Then the action executes, the loop moves on, and the text is released. It is never serialized. Nothing in the persisted record derives from it.

Two adjacent fields are absent for the same reason. The set of tools the agent actually considered before choosing is not captured, only the static list of tools available and the single tool selected. The rationale for rejecting the alternatives is not captured either. So the record contains the endpoint of a selection process and none of the process.

The consequence is stronger than a retention gap, and it is worth stating precisely, because the distinction determines what kind of fix is possible. If reasoning were written and then aged out, the answer would be to retain it longer. If reasoning were written to a system an auditor cannot reach, the answer would be to grant access. Neither is the case. The reasoning is never written at any point, by any component, to any store. Reconstructing why an agent chose a particular action is therefore impossible after the fact by construction, not by oversight. No amount of log analysis, correlation, or forensic effort applied to the existing record can recover a fact that was never recorded. Figure 1 shows the exact point in the loop at which the information is lost.

That structural claim is what motivates this paper, and it has a corollary that shapes the design. If the reasoning cannot be recovered later, it must be captured earlier, specifically before the action executes rather than after it returns. Any instrumentation that runs after the action is already too late, because the artefact it would need is gone. This single ordering constraint is why the proposal in Section 7 places its first insertion between precondition evaluation and action execution rather than alongside the existing log write.

The paper makes four contributions.

First, it specifies a decision record for autonomous agent actions: twenty-two fields covering identity, trigger, retrieved context, system state read at decision time, alternatives considered, selection rationale, confidence, outcome, downstream effect, human review disposition, and retention class.

Second, it classifies every one of those fields against a real production system as EXISTS, PARTIAL, or PROPOSED. That classification is the paper's most transferable result, because it separates what an operator already has from what genuinely requires new engineering. Twelve of the twenty-two fields already exist. Three are partial in ways that are analytically interesting, discussed in Section 5.3. Seven do not exist at all.

Third, it locates the change precisely. In a twelve-step path from trigger to stored record, ten steps already run in production. Two insertions are proposed, and the ordering constraint above dictates where they go.

Fourth, it assesses whether the instrumentation is affordable, using modelled estimates with stated derivation methods, and concludes that at the measured invocation rate and at a projected rate an order of magnitude higher the storage cost is under one megabyte per month.

What this paper does not do should be equally clear. The proposed schema is specified, not implemented. Nothing proposed here has been built, so nothing proposed here has been measured, and no evaluation of the proposal is offered or implied. Section 11 states the limitations in full.

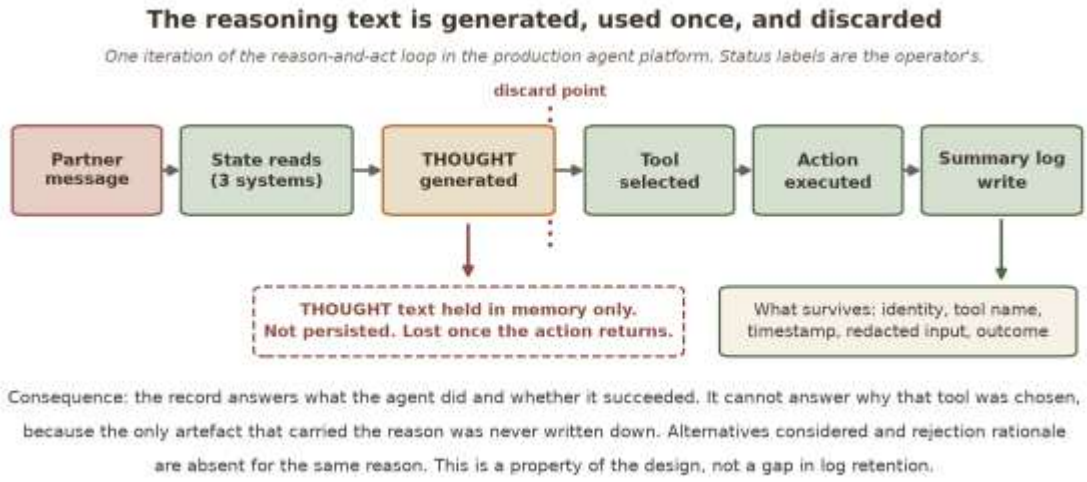


Figure 1. One iteration of the reason-and-act loop in the production agent platform, showing the point at which the reasoning text is discarded. All elements shown are current behaviour, status EXISTS as classified by the operator, except that the reasoning text is explicitly not persisted. No number appears in this figure.

2. Background and Prior Contribution

2.1 Agents that act

The reason-and-act paradigm interleaves natural-language reasoning with tool invocation, so that a model alternates between producing a thought about what to do and taking an action in an environment [1]. Tool use has since been trained directly into models [2] and the resulting agent architectures have been surveyed extensively [3]. The pattern's practical appeal is that the intermediate reasoning improves action selection. Its relevance here is different and less often noted: the paradigm produces, on every single decision, a natural-language artefact that states the basis of that decision. That artefact is exactly what an auditor of an autonomous action would want. In the standard implementation it is treated as scratch space.

There is a caution attached to that artefact which the rest of this paper takes seriously. Stated reasoning is not guaranteed to be the true cause of a model's behaviour. Chain-of-thought explanations can be systematically unfaithful, presenting a plausible justification that does not reflect the factors actually driving the output [4]. Self-reported confidence is likewise imperfectly calibrated, though it can be elicited more reliably with care [5]. Persisting the reasoning trace therefore buys a record of what the system stated as its basis for acting. It does not buy a certified causal account. Section 11 returns to this, because it bounds what the proposal can honestly claim.

2.2 Observability and audit logging

Distributed tracing established that causally linked events across a service estate can be captured with low overhead and sampled selectively [8, 9]. That lineage explains both the strength and the shape of the gap. Tracing answers questions about control flow and latency across components. It has no representation for a choice made inside one component between several permissible actions.

The audit-logging literature contributes the integrity requirements. Secure audit logs were formalised around the property that entries written before a compromise remain trustworthy after it [10]. Certificate Transparency demonstrated tamper-evident append-only logging at internet scale [11], and storage-level designs continue that line for general-purpose logging [12]. Provenance research supplies the complementary vocabulary of derivation: recording not only a result but the chain of inputs and processes that produced it [13].

The accountability literature supplies the requirement in its strongest form. Reviewable automated decision-making argues that a system is accountable only when a record exists that allows a decision to be reviewed after the fact, and that such records must be designed in rather than retrofitted [14]. Post-hoc explanation of a black-box model is a weak substitute for a decision that was recorded as it was made [15, 16]. The broader accountability literature frames the same requirement for algorithmic decision systems generally, treating the availability of a reviewable record as the precondition for holding anyone answerable [18]. Governance frameworks now state traceability and record-keeping as expectations for AI systems in consequential settings [17]. Human oversight of automation has its own long-standing framing, where the level of automation determines what the human retains and therefore what must be presented to them [7]. The agent platform described here sits deliberately at a mid-level: it acts autonomously for reversible work and queues production-affecting actions for human approval.

The gap this paper addresses sits between these bodies of work. Tracing gives the transport and low-overhead capture. Audit logging gives the integrity model. Provenance gives derivation. Accountability gives the requirement. What is missing is the record itself: a concrete field-level specification of what to store when the entity taking the action is probabilistic, grounded in what a production system can actually supply.

2.3 Contribution boundary against prior published work

The author has previously published on anomaly detection and root-cause synthesis for cloud infrastructure operations [19]. That work addressed the detection problem. It combined statistical detectors with log correlation and a language model that synthesised root-cause guidance, deployed on a managed Kubernetes service, and it reported a reduction in time to first actionable insight together with accuracy ratings on the generated analyses.

The boundary between that contribution and this one is worth stating explicitly, because both involve language models applied to infrastructure and the distinction is easy to blur. The prior work is about the system's own faults. It takes telemetry that already exists, notices that something is wrong, and proposes why the infrastructure failed. The agent is an observer of the estate and produces a diagnosis for a human to act on.

This paper is about the system's own actions. Its subject is not a fault but a choice, and not the infrastructure's behaviour but the agent's. It asks what record must exist so that a decision taken autonomously by the agent can be reviewed afterwards, and it answers with a field-level schema, a status classification against production, and a pipeline placement. Decision-level observability of agent actions is the contribution; anomaly detection and root-cause synthesis are established in the prior work and are not restated as new here.

One quantitative result from the prior work is used in this paper and is attributed to it rather than presented as new. That work established a confidence signal on its root-cause synthesis output, with 80 to 85 percent accuracy correlated with a HIGH confidence rating [19]. That figure is REAL and previously published. It matters here for a specific reason discussed in Section 5.3: it demonstrates that a usable confidence signal on a

language-model output is achievable in this operator's environment, which is why the absence of any equivalent signal on the action-selection path is a gap rather than an impossibility. The measured invocation baseline of 269 tool invocations over six months, roughly 45 per month, is likewise REAL and drawn from that published production data.

3. System Context: A Production Agent Platform Under Human Gating

The deployment is a partner-onboarding agent platform. To avoid naming internal or proprietary systems, this paper uses the operator's functional names throughout: Identity Service for the identity provider, Gateway for the API gateway, Secrets Store for the secrets store, Ticketing System for the ticketing system, Onboarding Agent for the agent platform itself, and Knowledge Index for the vector index over documentation.

The platform has been in production since 2024. The agentic phase, in which the platform selects and invokes tools rather than following a fixed script, has been active for roughly twelve to eighteen months. All of the following describes current behaviour with status EXISTS.

A partner sends a message in natural language through a portal. The Onboarding Agent queries the state of three external checks: registration status, IP allowlist status, and risk assessment status. It retrieves relevant documentation from the Knowledge Index. It then selects one of four tools: documentation question answering, IP allowlist request, service ticket creation, and credential generation. Credential generation is the consequential one, and it is bounded by an environment enum with two values, NP and PD, which separates the non-production and production environments and gates which actions are permissible in each.

Three existing controls shape what auditability is already possible.

Preconditions are evaluated as explicit gates before an action runs, and each gate's name and boolean result are logged per tool. A credential generation request, for instance, is gated on registration being approved, the relevant allowlist being complete, and the risk assessment being complete. A failed gate blocks the action.

Actions that affect production are queued for human review rather than executed directly, and the disposition of that review is recorded as approved, rejected, pending, or not applicable. The platform is therefore not fully autonomous by design. It sits at a level of automation where a human retains authority over production-affecting change [7].

Every action write is append-only and bound to a verified identity, discussed in Section 8.

This combination is what makes the deployment a useful subject. The controls an accountability framework would ask for at the action level are largely present. The gap is not a lack of engineering discipline. It is that the record those controls produce is an action record, and an autonomous agent needs a decision record.

4. Why Post-Hoc Reconstruction Fails Today

Section 1 stated the structural claim. This section establishes it field by field, because the claim is only as strong as the enumeration behind it.

Suppose an operator wants to answer the following question about an action taken three weeks ago: why did the agent generate non-production credentials for this partner rather than answer their question from documentation, given that the risk assessment was incomplete at the time?

From the existing record, the following is available. The decision identifier, the timestamp, the session identity, the environment, the triggering message in scrubbed form, the retrieved context identifiers and similarity scores, the values read from each external system with their read timestamps, the static list of tools available, the tool actually selected, the precondition gate results, the outcome, and the human review disposition where applicable. That is a substantial record and it settles a great deal. It establishes that the action occurred, who it was for, what the world looked like at the moment of decision, that the gates passed, and what happened next.

It cannot settle the question asked. Four specific absences block it.

The reasoning trace is absent. The natural-language statement in which the agent set out its basis for acting was generated and discarded, so the agent's own account of its decision does not exist in any store.

The alternatives considered are absent. The record shows four tools were available and one was invoked. It does not show which of the four entered contention. A tool never considered and a tool considered and rejected are indistinguishable afterwards, and they are very different facts about a decision.

The rejection rationale is absent. Even where an alternative can be inferred to have been plausible, the basis on which it was set aside was not captured.

The classification of the decision is absent. Nothing in the record distinguishes a decision determined entirely by deterministic gate evaluation from one where a probabilistic component exercised judgement. An auditor cannot tell which decisions actually warrant scrutiny, which makes prioritising review across a population of decisions impossible.

The result is that reconstruction is not merely laborious. It is unavailable. The best an investigator can do is infer a plausible reason from the surrounding state, which is a reconstruction of what a reasonable agent might

have done rather than a record of what this agent did. That inference cannot be validated against anything, because the thing it would be validated against was never written.

This is the same objection that the interpretability literature raises against post-hoc explanation of model behaviour, arriving here from the systems side [15, 16]. A justification generated after the fact, whether by a human investigator or by a model, is not evidence about the original decision. The remedy in both literatures is the same: record the basis at decision time [14].

Two further absences degrade the record without blocking this particular question, and they are worth naming because they matter for a different class of failure. The state read at decision time is captured, but it is never compared against the state at any later moment, so a decision that was correct when made and wrong moments later, the signature of a race condition, leaves no trace of having been overtaken. Studies of large-scale service outages find that failures frequently arise from interactions between components and their state rather than from isolated faults [6], which is precisely the class of problem a state comparison would surface and a state snapshot cannot. And the effect of the action downstream exists only as free text in ticket systems, so the consequence of a decision cannot be joined to the decision programmatically.

5. A Decision Record Schema for Autonomous Actions

5.1 The record

Table 1 specifies the record. Each field carries the status the operator assigned to it against the production system: EXISTS means the field is written today, PARTIAL means the signal is present somewhere but not in a usable audit form, and PROPOSED means it does not exist in the current production system. Figure 2 presents the same classification as a map.

The design principle is that the record should answer three questions in sequence: what did the agent know, what did it choose and on what stated basis, and what happened as a result. The existing log answers the third well, part of the first, and none of the second.

Table 1. The decision record, twenty-two fields, with per-field status against the production deployment. Status values are the operator's own classification: EXISTS, PARTIAL, PROPOSED. No field in this table carries a measured quantity.

Field	Status	What it holds
decision_id	EXISTS	UUID v4, generated at decision start
timestamp	EXISTS	ISO 8601, UTC
session_identity	EXISTS	Hash derived from an Identity Service token claim, never user-supplied
environment	EXISTS	Enum: NP or PD
triggering_message	EXISTS	Verbatim partner message, scrubbed of personally identifiable information at write time
retrieved_context	EXISTS	Knowledge Index chunk identifiers with similarity scores
system_reads	EXISTS	List of source, value and read timestamp per external check
tools_available	EXISTS	Static list, four tools
tools_considered	PROPOSED	Not captured today; the reasoning text is generated but discarded after the action executes
tool_selected	EXISTS	Name of the invoked tool
rejection_rationale	PROPOSED	Not captured today
precondition_checks	EXISTS	Name and boolean result per gate, already logged per tool
reasoning_trace	PROPOSED	The reasoning text exists only in memory during the reason-and-act loop; not persisted
decision_type	PROPOSED	No current classification of deterministic versus probabilistic

confidence_explicit	PARTIAL	Only the root-cause synthesis tool emits this today; action tools do not
confidence_implicit_proxies	PROPOSED	Not captured today
system_state_at_decision	PARTIAL	Captured as system_reads, but not diffed against later state
state_vs_reality_delta	PROPOSED	Not captured today
outcome_status	EXISTS	SUCCESS or FAILURE
downstream_effect	PARTIAL	Free text in ticket systems only, not structured
admin_review_disposition	EXISTS	APPROVED, REJECTED, PENDING or N/A
retention_tier	PROPOSED	No tiering exists; a single flat retention applies today

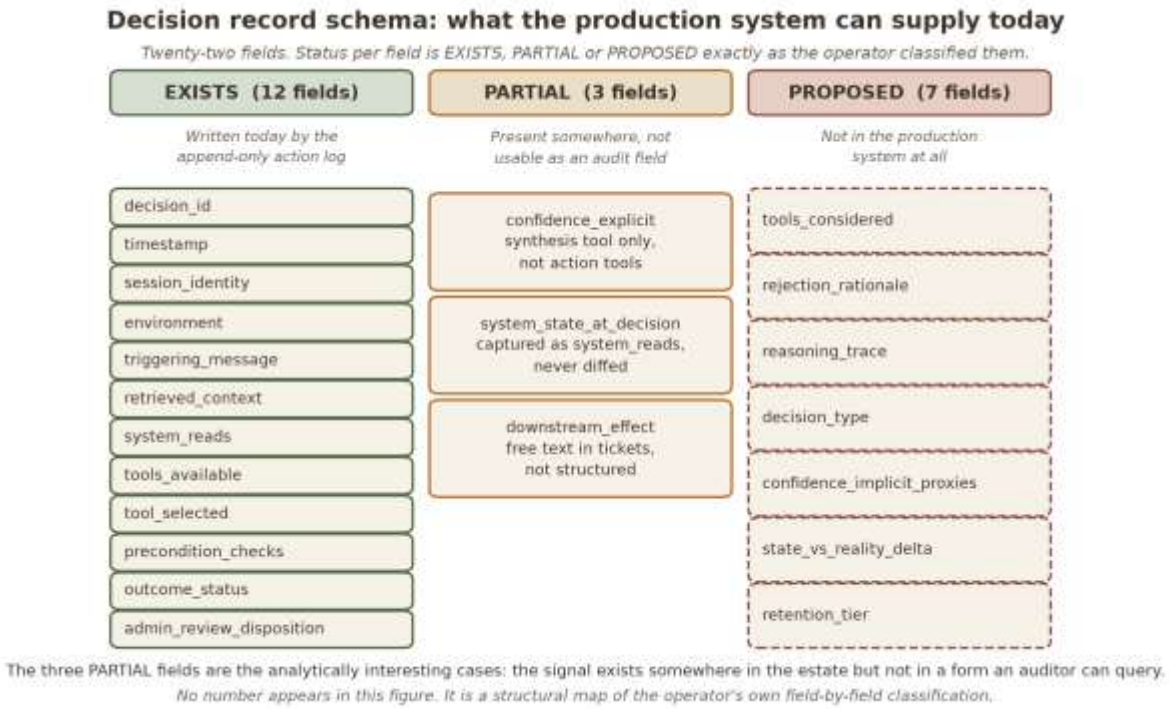


Figure 2. The twenty-two field record grouped by what the production system can supply today. Twelve fields EXIST, three are PARTIAL, seven are PROPOSED. Statuses are the operator's classification, not an assessment made in this paper. No number appears in this figure.

5.2 An illustration of the target schema

The listing below shows a populated record. It is a reconstructed synthetic illustration, not a captured production record, and it must be read as such for a specific reason: several of its fields, including tools_considered, reasoning_trace and rejection_rationale, do not exist in the current logs, so no production record containing them can exist. The illustration shows what the schema is for. It is not evidence that the schema has been populated.

```
{
  "decision_id": "8f2a41d0-synthetic-001",
  "timestamp": "2026-04-14T09:12:33Z",
  "session_identity": "sess-hash-7c19",
  "environment": "NP",
  "triggering_message": "Can you generate my API credentials
    for testing?",
  "retrieved_context": [],
  "system_reads": [
    {"source": "registration_status", "value": "APPROVED",
     "read_at": "09:12:31Z"},
    {"source": "ip_allowlist_NP", "value": "COMPLETE",
     "read_at": "09:12:31Z"},
    {"source": "risk_assessment", "value": "IN_PROGRESS",
     "read_at": "09:12:32Z"}
  ],
  "tools_available": ["credential_generation",
    "ip_allowlist_request", "service_ticket", "docs_qa"],
  "tools_considered": ["credential_generation", "docs_qa"],
  "tool_selected": "credential_generation",
  "rejection_rationale": {
    "docs_qa": "request is action-oriented, not
informational"
  },
  "precondition_checks": [
    {"name": "registration_approved", "result": true},
    {"name": "ip_allowlist_NP_complete", "result": true},
    {"name": "risk_assessment_complete", "result": false}
  ],
  "reasoning_trace": "NP allowlisting is complete and
    registration is approved. PD credentials are not
    appropriate yet because the risk assessment is still
    in progress. Safe to generate NP credentials only.",
  "decision_type": "hybrid",
  "confidence_explicit": null,
  "confidence_implicit_proxies": {"iteration_count": 2},
  "outcome_status": "SUCCESS",
  "downstream_effect": "NP client credentials issued via
    secure portal link",
  "admin_review_disposition": "N/A",
  "retention_tier": "full-trace-90d"
}
```


Listing 1. A populated decision record, marked SYNTHETIC. This is a reconstructed illustration of the target schema, not a production record, because tools_considered, reasoning_trace, rejection_rationale, decision_type, confidence_implicit_proxies and retention_tier do not exist in the current production logs.

Read against Section 4, the illustration shows what the additional fields do. The third precondition is false and the action still proceeds, which in the existing record would look like a gate failure followed by an action, an outcome an auditor should question. The reasoning trace resolves it: the incomplete risk assessment blocks production credentials specifically, and the agent scoped the action to the non-production environment for that reason. Two of the four tools entered contention and the rejected one carries a stated basis for rejection. The decision is classified as hybrid, indicating that deterministic gates and probabilistic judgement both contributed. None of that is inferable from the existing record.

5.3 The partial fields

The three PARTIAL fields deserve separate treatment, because they are the cases where the signal exists somewhere in the operator's estate but not in a form that can serve an audit.

confidence_explicit exists, but on the wrong system. The root-cause synthesis tool emits a confidence rating with its output, and that rating carries real information: 80 to 85 percent accuracy correlated with a HIGH rating, a REAL published result from the prior work [19]. The action-selection tools emit nothing equivalent. So the operator has demonstrated, in their own environment, that a usable confidence signal on a language-model output is achievable, and the system that takes consequential actions is the one that lacks it. That asymmetry is the argument for adding it. It also inherits the caution from Section 2.1: self-reported confidence is a signal to be calibrated against outcomes, not a guarantee [5].

system_state_at_decision is captured, as system_reads, but it is never diffed. The record preserves what each external check returned and when. It does not preserve whether any of those values changed between the read and the action. A decision that was correct at read time and stale by execution time is therefore indistinguishable from one that was correct throughout. This is the race-condition case, and it is the difference between storing a state snapshot and storing a state comparison. The proposed state_vs_reality_delta field is the comparison.

downstream_effect exists as free text in ticket systems. A human reading a ticket can see what an action caused. Nothing can join that consequence to the decision record programmatically, so the population of decisions cannot be analysed by outcome. Structuring this field is what would make the record usable for the trend analysis that the indefinite retention tier in Section 10 is meant to support.

6. What Exists Today Versus What Is Proposed

Stated plainly, and separated so that the boundary between the deployment and the proposal cannot be misread.

Exists today, status EXISTS:

- Append-only action-level audit logging, capturing identity, tool, timestamp, redacted input and outcome. REAL and previously published, covering 269 invocations over six months [19].
- Precondition check evaluation and logging, per gate and per tool.
- Admin-review gate and disposition tracking for production-affecting actions.
- A confidence signal on the root-cause synthesis tool only, which is a separate system and not the onboarding agent. REAL and previously published, 80 to 85 percent accuracy correlated with HIGH confidence [19].

Does not yet exist, status PROPOSED:

- Reasoning-trace persistence. The reasoning text is discarded after the action executes.
- Alternatives-considered and rejection-rationale capture.
- A confidence signal on the action-selection tools.
- System-state-delta detection, which is what would make race conditions visible.
- Tiered retention by risk level.
- A live full-decision-record schema at any production volume.

The last item is the one that bounds every claim in this paper. The schema in Section 5 has not been populated in production at any volume. It is a specification.

7. The Decision-to-Record Pipeline

Figure 3 sets out the twelve-step path from trigger to stored record, with each step tagged EXISTS or PROPOSED. Ten steps run in production today. Two are insertions.

Step 6, structured capture, serializes the reasoning trace, the alternatives considered, and the confidence proxies. Its position is the entire point. It sits after preconditions are evaluated and before the action executes. Placing it there follows directly from the structural argument in Section 1: the reasoning text is available at that moment and unavailable afterwards, so instrumentation that runs after the action returns cannot capture it at any cost. The ordering constraint is not a performance optimisation. It is a correctness condition on the record.

Step 10, the full decision record write, sends the complete record to the audit store tagged with a retention tier by risk level. It is deliberately separate from step 9, the existing action-level log write, and both are retained. Separating them keeps the existing append-only action log unchanged as the integrity-critical spine of the audit trail, and adds the richer decision record as a distinct write that can fail, be sampled, or be tiered without weakening the guarantee the action log already provides. Step 10 is also where the write can be made asynchronous, which is what keeps it off the latency path as discussed in Section 9.

The remaining ten steps require no modification, which is the practical claim of this section. The proposal is not a re-architecture of the agent platform. It is two insertions into an existing flow, one of which has a hard ordering requirement.

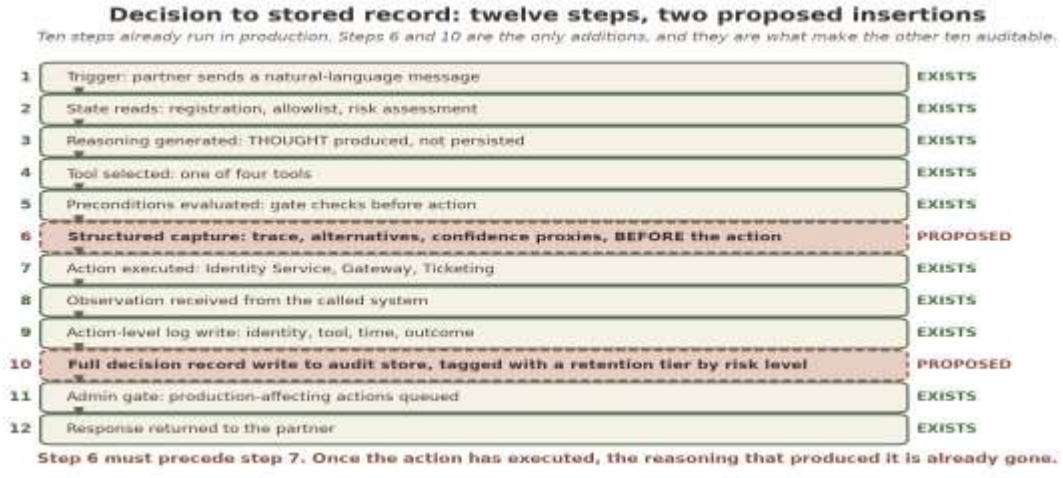


Figure 3. The twelve-step decision-to-record pipeline. Ten steps carry status EXISTS and two carry status PROPOSED, per the operator's classification. Steps 6 and 10 are the proposed insertions. No number in this figure is a measurement.

8. Identity Binding and Input Handling

Two properties of the existing write path are worth isolating, because they are already correct and because adding fields to the record must not disturb them. Figure 4 shows both.

The first is identity binding. The session_identity field is a hash derived from a token claim issued by the Identity Service. It is never user-supplied. The distinction is the whole security property. If the identity in an audit record were taken from a value the requester could set, then the record would attest to a claim rather than to a fact, and any entry could be repudiated by disputing the identifier. Because the value is derived from a verified token claim at the boundary and hashed before storage, the subject of a record cannot choose the value that names them. This is the property that makes the log evidence rather than testimony, and it is a precondition for the tamper-evidence guarantees that append-only logging provides [10, 11, 12].

Hashing rather than storing the claim directly is what allows that binding to coexist with data minimisation. The record retains a stable identifier that can be correlated across decisions and resolved back to a principal through the Identity Service when an investigation requires it, without the audit store itself becoming a directory of partner identities.

The second is input handling. The triggering_message field holds the partner's message verbatim, scrubbed of personally identifiable information at write time. Write time matters. Scrubbing on read, or on export, leaves the unscrubbed text at rest in the audit store, where it accumulates and where the population of readers over the record's lifetime cannot be predicted. Scrubbing before the write means the sensitive form never lands. For a record class whose retention is measured in months or is indefinite, that ordering is what makes the retention proposal in Section 10 defensible, and it reflects the minimisation principle that collection and retention should be limited to what the purpose requires [20].

The environment enum, NP or PD, completes the picture. It is not only a label on the record but a gate on what actions are permissible, so the record and the control share a field. An auditor reading a decision therefore reads the same value that constrained it.

The relevant conclusion for this paper is a negative one, and it removes an objection to the proposal rather than adding a capability. Every field added in Section 5 is either derived from the agent's own reasoning or from state the agent already read. None of them is supplied by the requester, and none of them changes how identity enters the record or when scrubbing occurs. The proposal enriches the payload of the write. It does not touch the write path's trust properties.

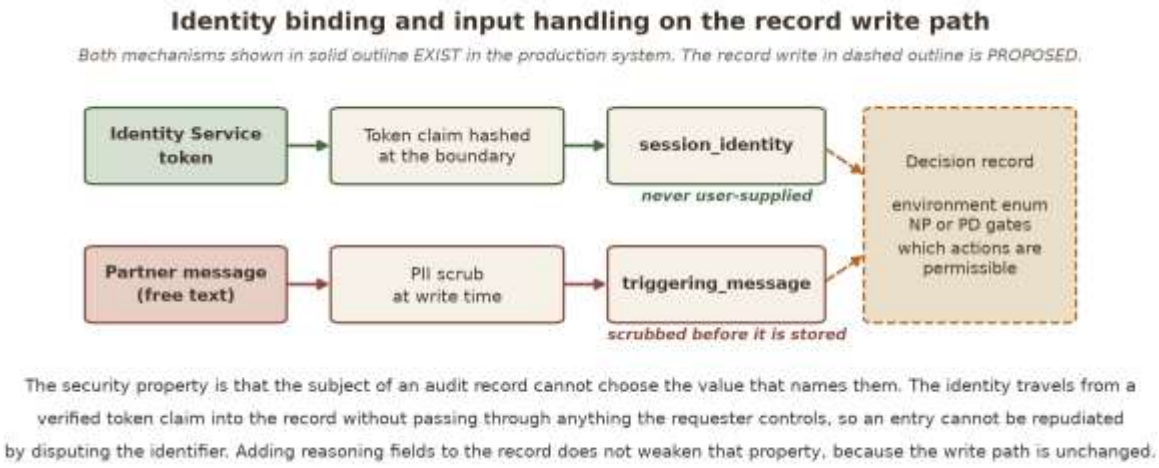


Figure 4. Identity binding and input handling. Both mechanisms shown in solid outline carry status EXISTS in the production system; the enriched decision record shown in dashed outline is PROPOSED. No number appears in this figure.

9. Instrumentation Overhead and Feasibility

The objection an operator will raise first is cost. Capturing the reasoning trace on every decision means additional generated tokens, additional latency, and a record roughly an order of magnitude larger than the summary it supplements.

Table 2 gives the estimates. Every figure in it is a modelled estimate rather than a measurement, and the derivation method is stated alongside each so that a reader can judge the estimate rather than accept it. Nothing proposed here has been implemented, so nothing in this table could be a measurement.

Table 2. Instrumentation overhead. All figures are SYNTHETIC modelled estimates, not measured production values. The derivation method is stated for each so the estimate can be assessed independently.

Metric	Estimate (SYNTHETIC)	Method of derivation
Added latency per decision	~180 to 260 ms	Estimated from structured-output token overhead at typical inference speed: roughly 40 to 60 added tokens at roughly 3 to 4 ms per token
Added token cost per decision	~55 tokens on average	Comparison of a structured reasoning template against free-form reasoning text, sampled across the four tool types
Record size on disk	~2.1 KB on average	Full record as JSON in UTF-8, against roughly 180 bytes for the existing summary-only record, roughly 11 to 12 times larger
Write path overhead	~8 to 15 ms	Asynchronous append to the audit store, isolated from inference time; modelled on comparable append-only log write benchmarks

Three observations follow, and they should be weighted differently because their inputs differ in quality.
The latency estimate is the softest figure in the table and it is the one an implementer should measure first. It is derived from a token count multiplied by a per-token rate, both approximate, so it inherits the error of both. What can be said with more confidence than the number itself is where the cost lands: on the inference path of a decision that already involves at least one model call and several external system reads, and only on decisions where the structured template is requested.
The write path cost is more robust, because it rests on an architectural choice rather than an estimate. The record write is asynchronous and outside inference time. A synchronous audit write in the request path would be a different and worse proposal, since it would make the agent's availability depend on the audit store's.
The storage arithmetic is the strongest of the three, because it is arithmetic over a stated policy. Full-trace capture is not applied to every decision. It is applied only to resource-modifying actions, which is the credential generation class in both environments. Under the projected volume scenario in Table 3, that class accounts for 190 invocations per month, giving roughly 190 multiplied by 2.1 KB, or about 400 KB per month of full-trace records. The summary record, at roughly 180 bytes, is written for every decision, giving the projected 598 decisions multiplied by 180 bytes, or about 108 KB per month. Combined, that is under 1 MB per month, which is negligible against any enterprise log retention budget.
That conclusion is robust in a way worth making explicit, because its robustness does not depend on the projection being accurate. Volume enters the calculation linearly. At the REAL measured baseline of 269 invocations over six months, roughly 45 per month, the total is an order of magnitude smaller again. At ten times the projected scenario it would still be under 10 MB per month. The storage argument therefore holds across the entire plausible range, and no volume assumption is load-bearing.
Table 3 states the projection used above. It requires an unambiguous label, because confusing it with production data would misrepresent the deployment.

Table 3. SYNTHETIC projected volume scenario, illustrative period of six months at roughly 600 invocations per month. These figures are NOT measured production data. They are an illustrative higher-volume projection used solely to exercise the scaling arithmetic in this section. The REAL measured baseline for this deployment is 269 tool invocations over six months, roughly 45 per month, from previously published production data [19].

Tool	Projected invocations over 6 months (SYNTHETIC)	Projected success rate (SYNTHETIC)
Documentation question answering	2,850	98.9%
IP allowlist request	210	93.8%
Service ticket	340	99.4%
Credential generation, non-production	118	100%
Credential generation, production	72	99.1%

Total (projection, not a measurement)	3,590, roughly 598 per month	not applicable

To restate the distinction one final time, because it is the single most consequential reading error a reviewer could make. The measured production volume for this deployment is 269 tool invocations over six months, roughly 45 per month, and that figure is REAL and previously published [19]. The 3,590 total and the roughly 598 per month rate in Table 3 are a projected higher-volume scenario, generated to demonstrate scaling behaviour. They are not measurements of this or any system, and they should never be cited as production volume.

10. Retention Tiering by Risk

The production system applies a single flat retention policy to all records. Tiering does not exist, and the retention_tier field carries status PROPOSED. Figure 5 sets out the proposal.

The rationale for tiering is the size asymmetry established in Section 9. A full-trace record is roughly eleven to twelve times the size of a summary record, and its value decays much faster. The detailed basis for a decision is what an investigation needs, and investigations occur within a bounded window after the event. The fact that a decision occurred, and its outcome, retains value indefinitely for trend analysis. Retaining both classes for the same period is the wrong trade in both directions: it either over-retains the expensive record or under-retains the cheap one.

Three tiers follow, all PROPOSED.

The full reasoning trace is retained for 90 days. The period is chosen to cover the typical incident-investigation and internal-audit review window, which is when a decision's detailed basis is actually needed.

The summary decision record is retained indefinitely in a low-cost tier. At roughly 180 bytes per record it is cheap enough that deletion saves little, and it is sufficient for long-term analysis of decision populations and outcome trends. This is the tier that the structuring of downstream_effect, discussed in Section 5.3, would make genuinely useful.

The admin-gated action record, covering access-control decisions that passed through human review, is retained on a longer horizon. Here the paper states a boundary rather than a number. A period in the range of a few years is consistent with common enterprise records-retention practice for access-control decisions, but the correct period is not a technical finding and this paper does not assert one. It must be set against the operator's own applicable records-retention obligation, which varies by jurisdiction, sector and the nature of the data involved. Operators adopting this schema should treat the tier as a parameter to be determined by their compliance function, not as a value inherited from a paper.

Two properties of the design are worth noting. Tiering is assigned at write time, as part of step 10, rather than applied by a later sweep, so a record carries its retention class from creation and the classification decision is itself auditable. And because the tier is keyed to the risk class of the action rather than to the requester or the outcome, retention cannot be influenced by the party whose actions are being recorded.



Figure 5. Retention tiering by risk class. All three tiers carry status PROPOSED; no tiering exists in the production system, which applies a single flat policy. The periods shown are the operator's proposal, not a regulatory finding, and the admin-gated tier must be set against the operator's applicable records-retention obligation.

11. Limitations

The limitations here are substantial and they bound the claims directly.

The schema is specified but not implemented or evaluated. No part of the proposed record has been built into the production system. There is therefore no measurement of it, no evaluation of whether the captured reasoning traces prove useful in a real investigation, and no evidence about how an auditor would actually work with the record. Every claim about the proposal is a design claim.

The only measured invocation figure is 269 tool invocations over six months, roughly 45 per month, drawn from previously published production data [19]. Every higher volume in this paper, including the 3,590 total and the roughly 598 per month rate in Table 3, is a projection generated to exercise the scaling arithmetic. No projected volume is a measurement of any system.

The overhead figures in Table 2 are modelled estimates with stated derivation methods, not measurements. The latency estimate in particular rests on two approximate inputs and should be measured before it is relied on for capacity planning.

The work describes a single platform in a single organisation, with four tools, one partner-onboarding workflow, and one human-gating arrangement. The field classification in Table 1 is the operator's assessment of that system. How much of it transfers to agents with larger tool sets, multi-agent delegation, or no human gate is untested.

No controlled comparison exists between the current instrumentation and the proposed instrumentation. The claim that the proposed record enables reconstruction where the current record does not is an analytical argument from the field enumeration in Section 4, not an experimental result. A stronger form of this work would run both instrumentations over the same decision population and measure whether investigators reach correct conclusions more often with the richer record.

A persisted reasoning trace records what the agent stated as its basis for acting. It does not certify that this was the true cause of the action. Stated reasoning can be unfaithful to the process that produced the output [4], and self-reported confidence is imperfectly calibrated [5]. The proposal should therefore be understood as making the agent's account of itself durable and reviewable, which is a real improvement over having no account at all, rather than as producing a verified causal explanation. Any confidence field added under this schema needs calibration against outcomes before it carries weight in a review.

Finally, the proposal increases what is retained about partner interactions. Scrubbing at write time and tiered retention are the mitigations, but a richer record is a larger surface, and the minimisation trade-off deserves explicit treatment by any operator adopting it [20].

12. Conclusion

The audit logging that autonomous agent platforms inherit records actions. When the entity taking the action selects it at run time, an action record is the wrong record, and the gap is not one of retention or access but of construction: the reasoning text that carries the basis for the decision is generated, used to select a tool, and released. Reconstructing why an agent acted is impossible afterwards because the artefact was never written.

The remedy is a decision record rather than an action record, and this paper specifies one. Its most useful result for other operators is not the field list but the classification behind it. Against a production agent platform, twelve of the twenty-two fields already exist, three are partial in ways that reveal where signals sit in the wrong place or the wrong shape, and seven require new capture. In a twelve-step pipeline, ten steps are unchanged and two insertions carry the proposal, with the first constrained to run before the action executes because afterwards is too late. The existing trust properties of the write path, identity derived from a verified token claim and never user-supplied, and scrubbing performed before the write, are preserved untouched. The storage cost is under a megabyte per month under a projected volume an order of magnitude above the measured baseline, and the conclusion holds across the plausible range because volume enters linearly.

The schema is a specification and not a result. Implementing it, and then testing whether investigators actually reach better conclusions with the richer record than without it, is the work that would turn this design into evidence.

References

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv preprint arXiv:2210.03629, 2022. Presented at the International Conference on Learning Representations, 2023. DOI: 10.48550/arXiv.2210.03629

[2] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. Advances in Neural Information Processing Systems 36, pages 68539 to 68551, 2023. DOI: 10.52202/075280-2997

[3] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen. A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6), 2024. DOI: 10.1007/s11704-024-40231-1

[4] M. Turpin, J. Michael, E. Perez, and S. Bowman. Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting. Advances in Neural Information Processing Systems 36, pages 74952 to 74965, 2023. DOI: 10.52202/075280-3275

[5] K. Tian, E. Mitchell, A. Zhou, A. Sharma, R. Rafailov, H. Yao, C. Finn, and C. Manning. Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 5433 to 5442, 2023. DOI: 10.18653/v1/2023.emnlp-main.330

[6] H. S. Gunawi, M. Hao, R. O. Suminto, A. Laksono, A. D. Satria, J. Adityatama, and K. J. Eliazar. Why Does the Cloud Stop Computing? Lessons from Hundreds of Service Outages. Proceedings of the Seventh ACM Symposium on Cloud Computing, pages 1 to 16, 2016. DOI: 10.1145/2987550.2987583

[7] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 30(3), pages 286 to 297, 2000. DOI: 10.1109/3468.844354

[8] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Google technical report, 2010. Available at https://research.google/pubs/dapper-a-large-scale-distributed-systems-tracing-infrastructure/

[9] J. Mace, R. Roelke, and R. Fonseca. Pivot Tracing: Dynamic Causal Monitoring for Distributed Systems. Proceedings of the 25th Symposium on Operating Systems Principles, pages 378 to 393, 2015. DOI: 10.1145/2815400.2815415

[10] B. Schneier and J. Kelsey. Secure audit logs to support computer forensics. ACM Transactions on Information and System Security, 2(2), pages 159 to 176, 1999. DOI: 10.1145/317087.317089

[11] B. Laurie. Certificate transparency. Communications of the ACM, 57(10), pages 40 to 46, 2014. DOI: 10.1145/2659897

[12] E. Soriano-Salvador and G. Guardiola-Muzquiz. SealFS: Storage-based tamper-evident logging. Computers and Security, 108, article 102325, 2021. DOI: 10.1016/j.cose.2021.102325

[13] L. Moreau. The Foundations for Provenance on the Web. Foundations and Trends in Web Science, 2(2 to 3), pages 99 to 241, 2010. DOI: 10.1561/1800000010

[14] J. Cobbe, M. S. A. Lee, and J. Singh. Reviewable Automated Decision-Making: A Framework for Accountable Algorithmic Systems. Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pages 598 to 609, 2021. DOI: 10.1145/3442188.3445921

[15] Z. C. Lipton. The mythos of model interpretability. Communications of the ACM, 61(10), pages 36 to 43, 2018. DOI: 10.1145/3233231

[16] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5), pages 206 to 215, 2019. DOI: 10.1038/s42256-019-0048-x

[17] E. Tabassi. Artificial Intelligence Risk Management Framework (AI RMF 1.0). National Institute of Standards and Technology, NIST AI 100-1, 2023. DOI: 10.6028/NIST.AI.100-1

[18] N. Diakopoulos. Accountability in algorithmic decision making. Communications of the ACM, 59(2), pages 56 to 62, 2016. DOI: 10.1145/2844110

[19] B. Kasinadhuni. From Detection to Action: AI-Driven Anomaly Detection and Root Cause Synthesis for Cloud Infrastructure Operations. International Journal of AI, BigData, Computational and Management Studies, 7, pages 171 to 184, 2026. DOI: 10.63282/3050-9416.ijaibdcms-v7i2p126

[20] P. Ganesh, C. Tran, R. Shokri, and F. Fioretto. The Data Minimization Principle in Machine Learning. Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency, pages 3075 to 3093, 2025. DOI: 10.1145/3715275.3732195