

The Intent Graph: A Field-Level Schema for Identity, Delegation, and Retroactive Correction in Autonomous Agent Systems

Daramola Benjamin

Levered

contact via levered.us

Abstract

Current work on securing autonomous AI agents treats identity, delegation, and error correction as separate problems, addressed by separate standards. This paper argues they are the same problem observed at three different points in a task’s lifecycle, and that the underlying cause is a single, correctable defect in how task intent is represented: fields of a task are either explicitly constrained by the user or left open for an agent to decide, and no current schema distinguishes the two. We show, from primary sources, that the three IETF efforts closest to solving agent security (Web Bot Auth, built on RFC 9421; OAuth 2.0 Token Exchange, RFC 8693; and emerging token attenuation drafts) each stop at the boundary the next one would need to cross, and that this boundary is a schema gap rather than a cryptographic or capability gap. We introduce the Intent Graph, an addressable schema in which every task field is tagged declared or open, dependency between fields is tracked explicitly, and both authorization revocation and post-hoc outcome correction operate as the same graph traversal at different times. We describe the full system architecture, the correction mechanism as a composition of event sourcing, causal dependency scoping, and compensating transactions, and the conditions under which the causal model can and cannot be trusted. We further show that correction alone is insufficient once a field’s correction window, the interval between an incorrect value being committed and its consequence becoming unrecoverable, is shorter than the system can detect and respond within, and extend the schema with a required-declared designation and a process conformance record for exactly this case, grounding the approach in the established, empirically validated doctrine of process conformance review used in aviation and surgical safety. We close with the limitations of the approach, stated without qualification.

1 Introduction

An AI agent that books a flight, changes a calendar entry, or migrates a codebase is executing a task whose specification was never fully determined by the person who requested it. A user who says “book me a flight to Dubai on Emirates, window seat” has fully constrained three things and left an unbounded number of others, cabin class, baggage allowance, refundability, unconstrained. The agent must fill these gaps to act at all.

This paper’s central claim is that every failure mode current agent security work is trying to prevent, an agent exceeding its authority, an agent silently defaulting to the wrong choice, an outcome that cannot be corrected without redoing the whole task, traces to this single fact: task intent is captured at a coarser resolution than execution requires. When an agent fills an unconstrained field with authority it technically holds, the result is a security failure, a re-statement of the confused deputy problem [1]. When it fills the same kind of gap with a default that turns out wrong, the result is a correction failure. These are not two problems requiring two solutions. They are one representational defect with two failure surfaces.

The examples used to motivate this claim, commerce and code, are chosen for clarity, not for stakes. For AI systems to

matter at the scale their advocates claim, they must eventually operate in domains where a wrong decision cannot simply be corrected afterward, surgery and structural engineering foremost among them. Section 7 shows why the correction mechanism this paper otherwise relies on is insufficient there, and what has to be added.

Section 2 establishes why this defect now matters at internet scale, using current traffic measurement data. Section 3 walks through the three IETF standards efforts closest to solving agent identity and delegation, quoting their own charters and drafts to show precisely where each one’s scope ends. Section 4 introduces the Intent Graph schema. Section 5 describes the full system architecture and the lifecycle of a single task through it. Section 6 details the correction mechanism. Section 7 extends the schema to domains where correction arrives too late to matter, and grounds this extension in existing process conformance doctrine from aviation and surgical safety. Section 8 addresses the causal learning loop the correction mechanism depends on. Section 9 states the system’s limitations without hedging. Section 10 discusses related work and positions this proposal against it.

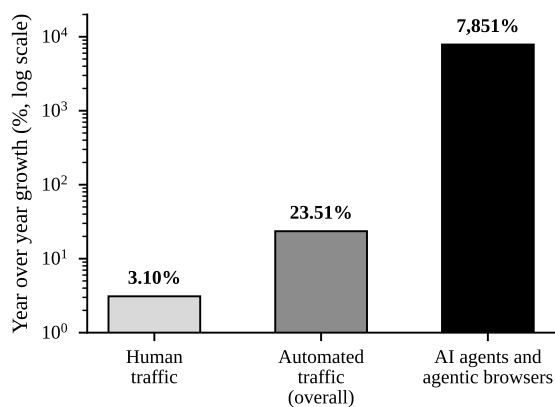


Figure 1: Year over year traffic growth by category, 2025, log scale. Source: HUMAN Security, 2026 State of AI Traffic and Cyberthreat Benchmark Report.

2 The traffic no current identity system was built for

Bot detection has historically relied on behavioral and structural signals that assume a human is not present: absence of mouse movement, mechanical timing between requests, the lack of a rendered viewport. This assumption has broken down. Computer-using agent products, including browser automation agents from major AI vendors, now read a rendered page and issue real pointer and keyboard events at the operating system level, rather than calling an API directly [9].

HUMAN Security's 2026 State of AI Traffic and Cyberthreat Benchmark Report, based on analysis of more than one quadrillion interactions across its Defense Platform in 2025, found that traffic attributable to AI agents and agentic browsers grew 7,851% year over year, while automated traffic overall grew 23.51%, against 3.10% growth in human traffic, a ratio the report describes as automation growing eight times faster than human traffic [9]. Figure 1 shows these three growth rates on a common log scale; the disparity in order of magnitude is the point, not an artifact of the scale choice.

The same report found that within agentic traffic, activity is no longer confined to passive browsing. Product and search pages account for the majority of agentic activity at 77%, but account pages (8.82%), authentication flows (4.95%), and checkout and payment flows (2.31%) indicate agents are now operating inside authenticated sessions and completing transactions autonomously (Figure 2) [9].

The structural problem this creates is shown in Figure 3. Traffic can be placed on two axes: whether it is indistinguishable from human traffic on the wire, and whether the authority behind it is borrowed from someone else rather than being the actor's own. A conventional API client (bottom left) is neither. A human (bottom right) is indistinguishable from itself by definition and acts on its own authority. A computer-using agent occupies the top right, indistinguishable from a human on the wire, while acting on authority delegated from

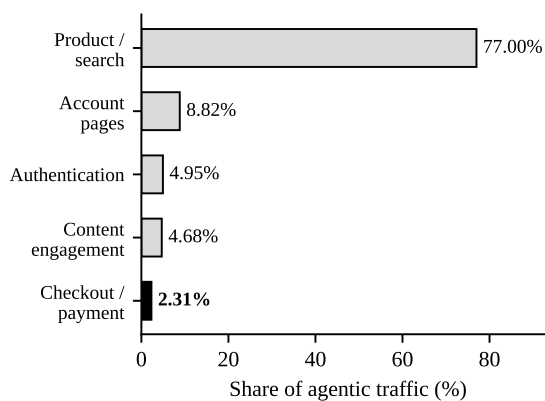


Figure 2: Agentic traffic by page category, 2025. Checkout and payment flows, while the smallest share, represent autonomous transaction execution without direct human involvement. Source: HUMAN Security, 2026 State of AI Traffic and Cyberthreat Benchmark Report.

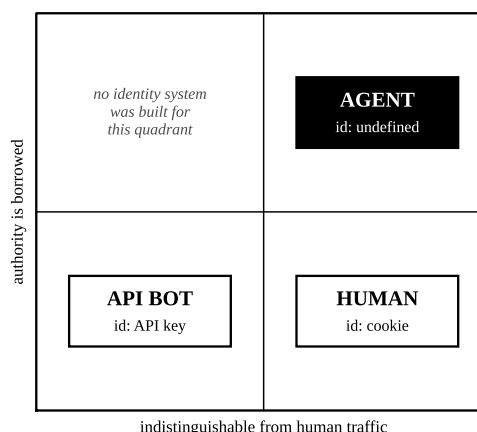


Figure 3: Traffic taxonomy by wire indistinguishability and authority origin. The agent quadrant is the one no current identity architecture targets.

someone else. No identity system in production today was designed for that quadrant, because until recently nothing occupied it at scale.

This reframes the design question. Detection, asking whether a given request is automated, was a tractable and largely solved problem as of roughly a decade ago. It is not a useful question when the automation is authorized, human sent, and indistinguishable from the human who sent it. The question that remains open is who authorized what, and how far that authorization reaches, which is a delegation and scope question, not a detection question.

3 Where the current standards stop

Three efforts under active IETF development address pieces of this question. Each is real, each is being implemented, and each explicitly bounds its own scope in a way that leaves the next layer's problem untouched.

3.1 Layer 1: agent identity is close to solved

Web Bot Auth, chartered as an IETF working group following a Birds of a Feather session at IETF 123, builds on RFC 9421, HTTP Message Signatures [2], to let an agent operator cryptographically sign outbound requests with an Ed25519 key, publish the corresponding public key at a well-known location, and let any origin verify that a request genuinely came from the claimed agent [4]. This is a real, deployable cryptographic proof of agent identity, and major infrastructure providers have already begun enforcing it at the edge.

The working group’s charter is explicit about what this does not cover. It states: “This effort will focus on authentication of the agent; authentication of the end user is out-of-scope” [4]. The charter separately lists “authenticating the end user of a participating client or agent” among its out of scope use cases. This is a deliberate, not accidental, boundary: the working group’s own minutes record a chair stating plainly that the end user is explicitly out of scope for the effort [5].

The consequence is precise. A verified Web Bot Auth signature tells an origin that a request came from, for example, a specific commercial agent product. It says nothing about which human’s session authorized the request, whether that human consented to the specific action being taken, or what limits they placed on it.

3.2 Layer 2: delegation has a chain, not a consent record

RFC 8693, OAuth 2.0 Token Exchange, defines a Security Token Service protocol supporting impersonation and delegation semantics, including a nested `act` claim that records, hop by hop, which actor is acting on behalf of which subject [3]. This is the closest existing standard to a delegation record for agent chains, and current proposals for agent-specific OAuth flows build directly on it [10].

Two properties limit its usefulness for the present problem. First, RFC 8693 does not define a mechanism for obtaining explicit, front channel user consent at the point an agent delegation is created; the token exchange endpoint consumes an already issued input token without specifying its provenance. Second, the specification’s own Section 4.1 states that nested `act` claims are informational only for access control purposes, not a cryptographically verifiable attenuation chain. A related weakness, termed “delegation chain splicing,” was raised on the IETF OAuth working group mailing list in March 2026: because RFC 8693 does not require cross validation between the `subject_token` and `actor_token` inputs, a compromised intermediary can in principle present mismatched tokens from different delegation contexts and receive a token asserting a chain that never actually occurred [7]. As of this writing the finding remains an open working group discussion, being routed toward a dedicated draft rather than a resolved specification change [8]; it is reported here as an active concern under discussion, not as a formally adopted working group position.

The chain RFC 8693 produces answers “who delegated to whom.” It does not answer “did the original subject agree to

Table 1: Coverage by standards layer.

Layer	Status	Gap
L1, agent identity (RFC 9421 / Web Bot Auth)	Solved	None; end user explicitly out of scope by design
L2, user delegation (RFC 8693)	Partial	Delegation chain recorded; consent not captured
L3, task scope (attenuation drafts)	Open	No standard object distinguishes declared from open task fields

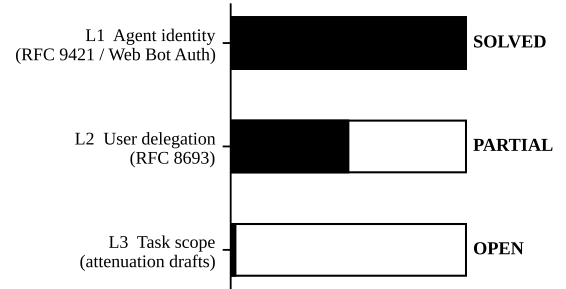


Figure 4: Coverage across the three identified standards layers.

this specific action,” because no consent record travels with the chain.

3.3 Layer 3: a scope is not a task

Even where a token is correctly scoped, a scope is a coarser object than a task. An OAuth scope such as `flights:write` permits every action the scope’s designer anticipated a legitimate client might need, checking availability, holding a seat, completing a purchase, because the scope has no field for what a specific user, on a specific occasion, actually intended. Emerging work, including draft `-niyikiza-oauth-attenuating-agent-tokens`, argues explicitly that RFC 8693’s nested `act` claims are informational rather than a self-verifiable attenuation chain, and that the specification defines no token-local mechanism for proving that a downstream delegation hop’s intent remains consistent with the original authorization [6]. This is precisely the confused deputy problem restated for delegation: a deputy holding broad, ambiently available authority cannot distinguish the specific purpose for which it was invoked from the full set of actions that authority would technically permit [1].

3.4 Summary: three layers, two schema gaps

Table 1 and Figure 4 summarize the state of coverage. None of the two open items are blocked on cryptography, protocol design, or engineering effort. They are blocked on the absence of a shared object that all three layers, and any correction mechanism built afterward, can reference. This is the gap this paper’s proposal addresses.

Table 2: Core fields of the IntentNode schema.

Field	Purpose
subject	User identity, consent timestamp, consent scope in the user's own terms
actor	Agent identity; signature and transport signals, satisfying Layer 1
task.declared	Fields the subject explicitly constrained
task.open	Fields left for the actor to decide; the locus of both authorization risk and correction need
depends_on	Explicit dependency edges between this node and others
outcome	What actually happened, per field, not as an unstructured result blob
outcome.diverged_from_declared	Fields where the outcome differs from what the subject declared, computed automatically
corrections	Append only log of correction events, each with its causal scope, compensations, and preserved fields

4 The Intent Graph schema

We propose a single addressable object, the IntentNode, referenced by every layer above and by the correction mechanism introduced in Section 6. Table 2 summarizes its structure.

The two fields that do the structural work are `task.open` and `outcome.diverged_from_declared`. Every other field already has a rough analogue somewhere in the standards reviewed in Section 3. These two do not. `task.open` is what makes a later correction legitimate, a field can only be retroactively changed if it was never declared, or if the subject is now amending a declaration. `outcome.diverged_from_declared` is what makes an error discoverable automatically rather than requiring the subject to notice it unaided, for instance by reading a bank statement.

5 System architecture

Figure 5 shows the full system as one continuous graph, not as separate modules. Every component reads from or writes to the IntentNode store.

Admission. Before an IntentNode is created for real, it passes through Composer, an admission layer that blends declared signals with passive transport signals, IP path, TLS or JA4 fingerprint, HTTP frame ordering, into a score. Declared identity is freely spoofable; transport level signals are comparatively expensive to fake at scale and form the root of trust the rest of the graph depends on. This is not an independent security feature. Every downstream component, outcome records, correction events, the causal model trained on both, is only trustworthy if admission-time identity is trustworthy; a compromised admission layer does not just create a security hole, it poisons every statistical model built on the graph afterward.

Orchestration. The admitted node is handed to a goal-directed execution layer, which resolves open fields using

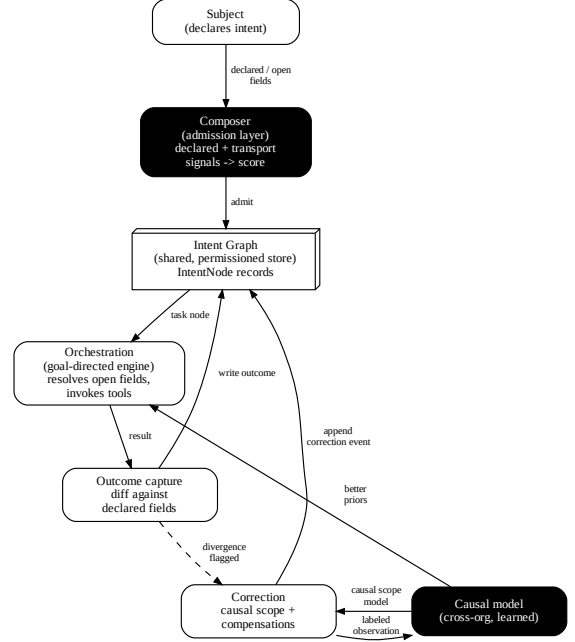


Figure 5: System architecture. The Composer and the causal model are the only components with independent write authority into the graph; every other stage passes through them.

whatever priors it has, invokes tools and third party APIs, and produces a result.

Outcome capture. The result is written field by field into `outcome`, and diffed automatically against `task.declared` to populate `diverged_from_declared`.

Correction. Detailed in Section 6.

Causal update. Every completed correction, whether it succeeded or failed, is written back into a shared causal model as a labeled observation, described in Section 7.

Figure 6 traces one IntentNode through this architecture end to end, using the running example introduced in Section 1: a user declares destination, carrier, and seat position, leaves cabin class open, and the agent defaults it incorrectly.

6 The correction mechanism

Retroactive correction is the least standardized part of this proposal and the part most existing systems have no equivalent for. We describe it precisely because the mechanism, while composed from known primitives, has not previously been specified for agent-executed tasks at the field level.

Given a correction request, for example, changing `cabin_class` from `first` to `economy` after a booking has completed, the system performs three steps.

1. **Causal scoping.** Using the causal dependency model described in Section 7, determine which other outcome fields are actually downstream of the field being corrected. For cabin class, this typically includes price and board-

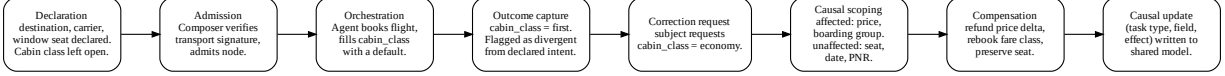


Figure 6: Lifecycle of a single IntentNode from declaration through causal update, worked through the flight booking example used throughout this paper.

ing group; it does not include seat position, travel dates, or passenger record identifiers. Fields determined not to be in scope are recorded explicitly as `preserved`, not merely left unmodified by omission.

2. **Compensation computation.** For each in-scope field, compute the minimal external action required to bring reality into agreement with the corrected value: refunding a price delta, rebooking a fare class, and so forth.
3. **Execution and append.** Attempt the compensations against whatever interface the third party, an airline, a SaaS vendor, a code repository, exposes for changes of this kind. Append the outcome of this attempt to `corrections` as an immutable event. The original outcome record is never rewritten, only patched forward, which yields a complete audit trail as a byproduct rather than as a separately engineered feature.

None of these three steps is a novel mechanism in isolation. Step three is event sourcing, treating state as an append only sequence of events rather than a mutable record. Step two is a compensating transaction, the pattern used in distributed systems to undo a partially completed multi-step action by executing its inverse rather than rolling back a database transaction. Step one is a scoped causal dependency graph, which is a direct extension, applied after execution rather than before it, of the fine grained controllability principle articulated for chip design and generative systems by Huang, who argued that the central unsolved requirement for agentic systems at every layer of the stack is the ability to change one precise element and have exactly its dependents update, rather than regenerating the whole [11]. The domains differ in an important way: a chip’s dependency structure is mechanical and known in advance, a compiler can enumerate exactly what a given net touches. A travel booking’s dependency structure is not mechanical. It must be learned from observed corrections, which motivates Section 7.

The contribution of this paper is not any one of these three mechanisms, each is established practice in its own subfield. The contribution is that composing them on top of an addressable, field-level intent schema is what makes minimal diff correction possible for tasks executed by autonomous agents, a capability no reviewed standard currently provides.

7 Process conformance and the correction window

The examples used so far understate what is at stake. A booking system and a code migration tool are useful because

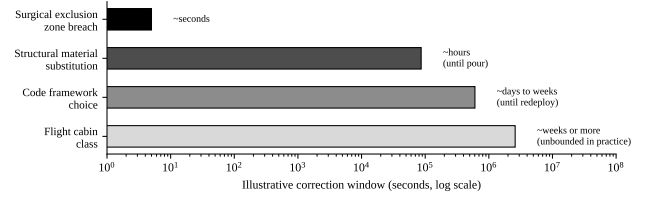


Figure 7: Correction window across four example fields discussed in this paper, illustrative and not measured, unlike the data reported in Figures 1 and 2.

they are simple to reason about, but for AI systems to affect economic output at meaningful scale, they must eventually operate inside the industries where decisions are hardest to reverse: surgery, structural and civil engineering, and other domains where an incorrect action can cost a life. These are also precisely the domains where the correction mechanism described in Section 6 does not help, because correction depends on an interval of time between a wrong decision and its consequence becoming unrecoverable, and in these domains that interval can be measured in seconds, or may not exist at all once an action is taken.

We formalize this interval as the *correction window* of a field: the time between an incorrect value being committed for that field and the point at which its consequence can no longer be undone or compensated. Cabin class has an effectively unbounded correction window, it can be changed indefinitely after booking. A framework choice in a code migration has a window bounded only by the next deployment. A material substitution in a structural design has a window that closes at the point of pour or fabrication. A surgical instrument approaching an anatomical exclusion zone has a window measured in the time it takes a monitoring system to detect the approach and a human to intervene, often seconds. Figure 7 places these four examples on one illustrative axis.

This reframes the design question raised in Section 6. Retroactive correction is a valid strategy exactly when a field’s correction window exceeds the system’s combined detection and response latency. When the window is shorter than that latency, correction cannot be relied upon as a safety mechanism, and the system’s obligation shifts from correcting an error after it occurs to preventing the error from occurring at all.

This is not a novel insight within these industries themselves. Aviation and healthcare have both formalized, over decades, a doctrine sometimes called just culture, in which safety is measured primarily by conformance to a defined

Table 3: Schema extension for domains with a short correction window.

Field	Purpose
<code>task.</code>	Fields the system may never default; unconstrained at execution time means orchestration halts rather than proceeds
<code>required_declared</code>	Append only record of which required fields were verified, by what mechanism, and when; reviewed independently of outcome
<code>process.conformance_log</code>	

process, evaluated independently of the outcome achieved, rather than by outcome alone. The FAA’s compliance philosophy states its objective as identifying the safety issues underlying a deviation from standards and correcting them, treating a deviation as reportable and worth investigating regardless of whether it happened to cause harm [15]. Hospital morbidity and mortality review operates on the same principle, examining whether care met accepted process standards independently of the clinical result obtained [16].

The empirical case for taking process conformance this seriously, rather than treating it as compliance overhead, is direct rather than inferred. Haynes et al. conducted a prospective trial of a 19 item surgical safety checklist across eight hospitals in eight cities, and found that checklist conformance alone, with no change to surgical technique, reduced inpatient mortality from 1.5% to 0.8% and major complications from 11.0% to 7.0% [17]. Process conformance did not merely correlate with better outcomes in this study. It was the intervention that produced them.

7.1 A required field, and a conformance record rather than a correction

The schema in Section 4 already distinguishes `task.declared` from `task.open`. We extend it with a third designation, summarized in Table 3: `task.required_declared`, fields the system must never resolve with an inferred or default value. If a required-declared field is unconstrained when execution would otherwise proceed, orchestration must fail closed, halt and demand explicit verification, rather than fail open and rely on Section 6 to repair the result afterward. Which fields belong in this set is not asserted per industry. It follows from the correction window directly: a field belongs in `task.required_declared` precisely when its correction window is shorter than the system’s detection and response latency.

The audit artifact this produces is structurally similar to, but functionally distinct from, the corrections log in Section 6. We term it `process.conformance_log`: an append only record, in the same event sourced style as `corrections`, but populated before or during execution rather than after, recording which required fields were verified, by what mechanism, and when. Critically, this record is reviewed independently of outcome, mirroring the just culture doctrine directly. A bad outcome reached through a fully conformant process is not, by itself, a failure of the system. A good outcome reached through a process that skipped a

required verification is.

7.2 Two worked examples

Surgical assist. A user’s declared intent specifies procedure type, target anatomical site, incision margins, and instrument force limits. Exclusion zones around critical structures, major vasculature, nerve bundles, are marked `required_declared`, because a default here has a correction window measured in the time before contact, not after. If intraoperative sensor data indicates the instrument is approaching a declared exclusion boundary, the system halts and requires explicit human confirmation before proceeding, rather than continuing and relying on Section 6 to log a compensating action afterward, because no such action exists once contact occurs. The resulting conformance log, not the surgical outcome alone, is what a hospital safety review examines.

Structural engineering. A design and build pipeline’s declared intent specifies load bearing requirements, material grade per structural element, and required safety factors. Any substitution touching the load path, for instance an automated procurement step selecting a cheaper beam grade, is `required_declared`. The correction window for this field closes at the point of pour or fabrication; after that point, correction requires demolition, not a database update. The system must re-verify any such substitution against the structural model and produce a signed conformance record before construction proceeds, which is the direct digital analogue of the professional engineer’s stamp already required in this field for exactly this reason.

7.3 What this does and does not solve

We are explicit that this section does not solve the actual bottleneck to AI operating in these industries. Liability assignment when an AI assisted decision contributes to harm, professional licensing frameworks, and regulatory clearance processes such as those governing surgical devices, are legal and institutional problems, not schema design problems. A conformance log of the kind described here is necessary evidence for any of those institutions to evaluate a claim. It is not sufficient to resolve any of them. What this section adds is a concrete, implementable answer to a narrower question this paper’s schema would otherwise leave open: how an AI agent’s decisions could be audited at all, by the same standard these industries already hold human practitioners to, in a domain where the correction mechanism in Section 6 provides no safety net.

8 The causal learning loop

Causal scoping in Section 6 requires knowing, in general and in advance of any specific correction, that cabin class does not affect seat assignment for a given carrier’s booking system. A single organization sees a limited volume of corrected task outcomes for any given task type, insufficient to learn this reliably in a useful timeframe. A shared graph, in which independent, non colluding organizations’ Composer

instances each contribute admitted, outcome tagged, and corrected IntentNodes to the same permissioned store, accumulates the relevant observations far faster, because the causal structure of a task type such as `book_flight` is shared across every organization that books flights, while any single organization’s correction volume is not.

Each completed correction, successful or not, is a labeled training example of the form (task type, field changed, fields actually observed to change with it). Accumulated across the shared graph, this produces two distinct outputs. First, a better prior before execution: if the causal model has observed that a given task type’s open field produces a correction request at a known rate, the orchestration layer can be instructed to surface the ambiguity to the subject before acting rather than defaulting silently. Second, a faster and more confident causal scope at correction time, since the scope in Section 6, Step 1 is drawn from an empirically supported model rather than guessed.

This is reinforcement learning in a precise, not metaphorical, sense: the policy is the choice made for an open field, the reward signal is whether that choice required a correction, and the causal graph performs credit assignment between a field level decision and its downstream consequences. It requires no simulated trials, since every real correction event in production is itself a training example.

Cold start is a genuine limitation, not a rounding error.

A task type or contributing organization new to the graph has no correction history. Until sufficient observations accumulate, causal scope cannot be trusted, and the system must fall back to the coarse behavior it exists to avoid, treating the entire task, rather than the individual field, as the unit of correction. This should be stated plainly to any evaluator of the approach rather than smoothed over in presentation.

9 Limitations

We state the following without qualification, because a system whose limitations are not stated as clearly as its capabilities is not evaluable.

1. **Third party cooperation is a hard external dependency.** The correction mechanism can construct a provably minimal compensation plan. It cannot compel a third party, an airline, a SaaS vendor, to execute it. The guarantee is bounded to the plan’s correctness, not to its real world completion.
2. **The causal dependency model’s reliability is domain dependent.** For code, where dependency is mechanically derivable from an import or call graph, the model is close to exact. For commerce and identity domains, it is statistical, inferred from observed corrections, and carries a real, non zero error rate that shrinks slowly with data volume rather than becoming exact.
3. **Cold start reproduces the original failure mode.** For any new task type or newly contributing organization, until sufficient corrected outcomes accumulate, the system

cannot offer field level correction and must fall back to coarse, whole task correction.

4. **The shared graph and the public aggregate view are in tension.** A cross organizational causal model requires broad, trustworthy contribution; any aggregate or public projection of the same graph, useful for showing builders what intents are being expressed at internet scale, requires strict controls on what is exposed outside the contributing organization’s boundary. The boundary between these two uses of the same underlying data has not been designed in this paper and is an open problem.
5. **This is a protocol proposal, not a deployed system.** The architecture in Section 5 has not been implemented at production scale. Claims in this paper about causal model accuracy and correction latency are architectural, not measured.
6. **The required-declared extension in Section 7 produces evidence, not institutional authority.** A conformance log is necessary material for a liability, licensing, or regulatory review in a high-consequence domain. It does not itself constitute clearance, certification, or legal protection, and building it does not shorten the institutional path to deploying an agent in surgery or civil engineering.

10 Related work and positioning

The confused deputy problem, as originally formulated by Hardy, concerns a deputy program that cannot distinguish which of its ambiently held authorities is relevant to a given caller’s request [1]. Recent work applies this framing directly to autonomous AI agents holding broad standing credentials across email, calendar, infrastructure, and code repositories [12], which is the security framing this paper adopts in Section 1 and extends with the correction framing in Section 6.

Work on capability based delegation, including macaroon style attenuable tokens and holder side scope narrowing, addresses part of Layer 3’s gap by letting a token holder narrow a grant without a round trip to an authorization server. Comparative analysis of OAuth 2.1 against capability based and macaroon based delegation notes that OAuth provides no holder side scope attenuation, only the authorization server can narrow scope at issuance time, and that OAuth tokens are opaque to intermediaries in a way that loses record of the original authorization chain [13]. The Intent Graph’s `task.open` versus `task.declared` split is complementary to this direction: attenuation narrows what a token permits, while the declared and open split records why a narrower or wider grant was appropriate in the first place, which attenuation mechanisms alone do not capture.

Independent proposals for delegation provenance records, carrying an append only chain of delegation history that can be verified offline, address part of Layer 2’s consent gap without requiring a reachable authorization server at verification time [14]. These are compatible with, rather than competing against, the architecture in Section 5; a provenance

record of this kind is a natural candidate implementation for the subject and actor fields of the IntentNode schema.

11 Conclusion

Identity, delegation, and correction for autonomous agents are usually treated as three problems because they surface as three different kinds of failure: a security incident, an unaudited grant of authority, and a wrong outcome nobody can fix without redoing everything. This paper has argued they share one cause, task intent expressed at a coarser resolution than execution requires, and shown that the standards closest to solving agent security each stop exactly at the boundary the next one needs to cross, not because the remaining work is hard cryptography, but because no shared, field addressable object exists for any of them to write into. The Intent Graph is a proposal for that object. Getting authorization right prevents an agent from doing what it should never have been allowed to do. It does not, by itself, fix what happens when an agent acts entirely within its authority and still gets something wrong. The same schema that governs the first problem, we have argued, governs the second, because they were never two problems to begin with. Section 7 extends this one step further: in domains where the correction window closes before a mistake can be caught, the same declared and open distinction becomes the mechanism for proving, in advance and independently of outcome, that a required step was not skipped, which is the only form this system’s guarantee can take once fixing a mistake afterward has been ruled out by physics rather than by policy.

References

- [1] N. Hardy, “The Confused Deputy (or Why Capabilities Might Have Been Invented),” *ACM SIGOPS Operating Systems Review*, vol. 22, no. 4, pp. 36–38, Oct. 1988. DOI: 10.1145/54289.871709.
- [2] A. Backman, Ed., J. Richer, Ed., M. Sporny, “HTTP Message Signatures,” RFC 9421, IETF, Feb. 2024.
- [3] M. Jones, A. Nadalin, B. Campbell, Ed., J. Bradley, C. Mortimore, “OAuth 2.0 Token Exchange,” RFC 8693, IETF, Jan. 2020.
- [4] IETF Web Bot Auth Working Group, “Charter for Working Group,” IETF Datatracker, chartered following BoF at IETF 123, 2025–2026.
- [5] “Minutes, IETF 124: webbotauth,” IETF Datatracker, Nov. 2025.
- [6] J. Niyikiza, “Attenuating Agent Tokens,” Internet-Draft, draft-niyikiza-oauth-attenuating-agent-tokens-01, IETF, Jun. 2026.
- [7] Chiradeep, “Security Consideration: Delegation Chain Splicing in RFC 8693 Token Exchange,” IETF OAuth Working Group mailing list message, Mar. 2026.
- [8] J. Lombardo, Reply thread, “Delegation Chain Splicing in RFC 8693 Token Exchange,” IETF OAuth Working Group mailing list, Mar. 2026.
- [9] HUMAN Security, Inc., “The 2026 State of AI Traffic and Cyberthreat Benchmark Report,” Mar. 2026.
- [10] WorkOS, “AI Agents and the Multi-Hop Delegation Problem,” WorkOS Engineering Blog, 2026.
- [11] J. Huang, remarks on agentic system controllability, Y Combinator Startup School, Jul. 2026.
- [12] Cloud Security Alliance Research, “Confused Deputy Attacks on Autonomous AI Agents,” CSA Research Note, 2026.
- [13] “AIP: Agent Identity Protocol for Verifiable Delegation Across MCP and A2A,” arXiv preprint, 2026.
- [14] “HDP: A Lightweight Cryptographic Protocol for Human Delegation Provenance in Agentic AI Systems,” arXiv preprint, 2026.
- [15] U.S. Federal Aviation Administration, “Compliance Program,” FAA Office of Safety Standards, faa.gov, 2026.
- [16] “Just Culture: A Foundation for Balanced Accountability and Patient Safety,” *PMC*, National Library of Medicine.
- [17] A. B. Haynes, T. G. Weiser, W. R. Berry, S. R. Lipsitz, A. H. Breizat, E. P. Dellinger, T. Herbosa, S. Joseph, P. L. Kibatala, M. C. Lapitan, A. F. Merry, K. Moorthy, R. K. Reznick, B. Taylor, A. A. Gawande, for the Safe Surgery Saves Lives Study Group, “A Surgical Safety Checklist to Reduce Morbidity and Mortality in a Global Population,” *New England Journal of Medicine*, vol. 360, no. 5, pp. 491–499, Jan. 2009. DOI: 10.1056/NEJMsa0810119.