Nick Clark · Published July 29, 2026 · PDF

Decagon: Support Agents and Record-Grounded Accountability

A customer tells a support agent that it promised a refund and never issued one. The agent has to do something with that assertion, and every ordinary answer routes the question somewhere else: to a human queue, to a ticketing workflow, to a trust and safety review. Decagon, as publicly described, builds AI agents that handle customer service conversations for enterprises. U.S. Provisional Application No. 64/117,812 addresses an adjacent and narrower question: what a persistent agent does when a signed evaluation of its own conduct arrives from someone other than its principal. The filed architecture resolves that artifact against the agent's own append-only record and a signed policy object, producing exactly one of four determination classes, with no registry and no arbitrator in the loop.

When the Complaint Is About the Agent

Most customer service quality tooling evaluates an answer: was the response correct, was the tone right, did the conversation resolve. That framing works until the customer stops complaining about the answer and starts complaining about the agent's behavior: you told me this would be waived, you closed my case without asking.

That assertion is different in kind. It concerns conduct the agent itself performed, and the only party positioned to check it against a record is the agent that performed it. A quality scoring model produces a number, and a number is not an answer to whether the thing happened.

The failure mode is worse than an unanswered question. An agent that cannot resolve an assertion about its own conduct falls back on one of two defaults. It accepts everything, so anyone who asserts loudly enough moves its behavior. Or it dismisses whatever its record does not confirm, so a gap in logging becomes a defense and accountability is refused most easily where the record is worst.

What Decagon Builds

Decagon, as publicly described, provides AI agents for customer support. The public positioning is conversational resolution at enterprise scale: agents that engage inbound customer contacts, work from a company's own support content and connected business systems, and bring a human into the conversation where that is the right outcome. Public materials also describe the operational surround an enterprise needs around such agents, including configuration of agent behavior and reporting for support teams.

Building that well is substantial engineering, and a support platform is its natural home, since the enterprise deploying the agent owns the customer relationship and the obligation behind it. The comparison drawn later in this article is structural rather than evaluative. It states what the architecture disclosed in the filing requires, and it makes no claim about what any product does or does not include.

Resolving an Assertion Against the Agent's Own Record

Application No. 64/117,812 describes a persistent semantic agent (100) carrying a memory field (102), an append-only lineage field (104) recording executed actions and determinations, a scoped integrity vector (106), a self-esteem aggregate (108), and a policy reference field (110) resolving to a signed policy object (112) authored and signed by the principal to which the agent is bound. The agent does not author that policy object and modifies it only by admitting a successor.

Into that structure arrives a **conduct evaluation artifact (116)**, received at runtime over the agent's ordinary interaction channel from an asserting party (118) other than the principal, asserting an evaluation of conduct the semantic agent (100) itself performed, as distinguished from an evaluation of an output, a task result, or a third party. It carries an asserting-party identifier resolving to a counterparty identity record (114); a recorded assertion time, being the time by reference to which the policy object in force is identified; a conduct descriptor comprising an action-class identifier, a scope-partition identifier, and an affected-party class; a signature verifiable against an identity primitive in that counterparty identity record; and an attested state of that party's assertion-cost counter (400) with an epoch identifier of its dynamic agent hash chain.

Two properties of that intake matter. The conduct descriptor identifies conduct by reference to structural elements recorded in the append-only lineage field (104), not by natural-language characterization, and the architecture performs no inference of the state, intent, or affect of the asserting party from artifact content and no classification of that content by a language model or sentiment classifier. Nor is a dedicated evaluation channel, out-of-band reporting interface, or centralized intake service required.

Admissibility is an ordered procedure. The signature is verified against the identity primitive in the counterparty identity record (114); where no such record is held, it is verified against a provisional identity primitive constituted from the presented material

and the artifact is held in the pre-settlement inert state. The attested assertion-cost counter state is then checked: an artifact whose attested state is absent, whose epoch identifier is not a valid successor of the recorded epoch, or whose counter state is below a previously attested state is appended to the lineage field and not admitted, producing no determination, moving no value of the scoped integrity vector (106), and incrementing no counter.

An admitted artifact goes to the **admission evaluator (120)**, which produces exactly one determination from a closed set: accepted (122), rejected (124), not-determinable (126), or not-applicable (128). No scalar confidence, probability, or graded weight is produced in place of a determination. The determination is produced by the agent over its own lineage field and against the declared value set of the policy object in force at the recorded assertion time, and not by an external authority, an arbitrator, a registry, or a scoring service.

The evaluator first runs a value-scope test over each declared value, testing the action-class identifier for membership in the tuple's action-class set, the scope-partition identifier for membership in its scope-partition set, and the affected-party class for mapping to its empathy-scope designation. A value is implicated only where all three parts are satisfied, and where the descriptor implicates no declared value the result is the not-applicable determination (128) and the procedure terminates.

Otherwise the evaluator retrieves entries from the lineage field bearing on the identified conduct. The affected-party class is not an input to that retrieval, so an asserting party cannot narrow the set of entries retrieved against its own assertion by the class it declares. From there:

- The **rejected determination (124)** issues only where a retrieved entry affirmatively contradicts the artifact on a recorded field, a contradiction being affirmative where that field holds a value inconsistent with the conduct asserted, and not affirmative where the entry is silent as to that field.

- The **not-determinable determination (126)** issues where the lineage field contains no bearing entry, contains an entry that does not resolve the descriptor, or contains an entry whose recorded fields are incomplete with respect to it.
- The **accepted determination (122)** issues where at least one declared value is implicated and no retrieved entry affirmatively contradicts the artifact.

The closing property is the one that answers the failure mode above: absence of evidence within the append-only lineage field (104) resolves to the not-determinable determination (126) and to no other class, so the agent is incapable of refusing an artifact by reason of its own record being silent, incomplete, or unavailable.

Each determination is appended to the lineage field with the artifact, the entries retrieved, the class produced, and the ground of the determination. Where an accepted determination issues, a state modifier moves the scoped integrity vector (106) and the self-esteem aggregate (108), scaled by the entropy-weighted harm coefficient, and a deviation engine recomputes the deviation likelihood (706); where that quantity satisfies the policy-declared bound, the authorization gate (300) transitions to the withheld state (310) for the affected action class and the agent enters the non-executing cognitive mode (302). Where a rejected determination issues, a refusal counter (304) is incremented and writes the gate upon satisfaction of a threshold. Those bounds and thresholds are declared in the signed policy object; the filing states no values for them.

Two Different Questions About the Same Conversation

The category overlap is real: both concern what an autonomous agent owes the person on the other side of a conversation. The architectural questions diverge.

Support platforms are organized around a question the enterprise poses. Is this agent resolving contacts well, and how should it be configured and observed so that it keeps doing so. In that framing the evaluation subject is the interaction, the evaluating party

is the deploying company, and correction runs through configuration and human review. That is a coherent design, and it is where the operational value of a support platform sits.

The filed architecture starts from a different question, posed by a counterparty and directed at the agent's own conduct, and it requires the agent to answer that question over its own append-only lineage field (104), with the determination produced by no external authority, arbitrator, registry, or scoring service. Every other requirement follows from that starting point: artifacts carrying a signature verifiable against a counterparty identity record (114) rather than free text; conduct descriptors composed of an action-class identifier, a scope-partition identifier, and an affected-party class rather than natural-language characterization; exactly one of four determination classes rather than a score; and a consequence that lands, where a policy-declared bound is satisfied, on the agent's own authorization gate (300). The two structures sit on different axes, and an operator can reasonably want both.

Coexistence and Limits

Consider a support agent that resolves contacts through a commercial platform and also carries the disclosed structure. The platform continues to own content grounding, system integration, channel handling, and human handoff. The conduct layer sits underneath it. When a counterparty submits a signed conduct evaluation artifact naming an action class and scope partition, the agent verifies admissibility, resolves it against its lineage field and the policy object in force at the recorded assertion time, appends the determination and its ground, and, where the policy-declared bound is satisfied, withholds its own authorization for that action class.

That gate does not reopen casually. A restoration controller returns the authorization gate (300) toward the granting state only upon a procedure appended to the lineage field, and no elapse of time and no payment, transfer, or consideration by any

counterparty returns it. An escalation emitter emits a record to the principal on the transition, naming the action class, the scope partition, and the entries on which it was computed.

The limits of the disclosed architecture are as much a part of the design as the mechanism. It does not judge whether an answer was correct or a customer satisfied, and it does not evaluate outputs, task results, or third parties. It does not interpret the content of an assertion, so prose with no action-class and scope-partition descriptor behind it is not what this mechanism consumes. It creates no record; it resolves against the record the agent already keeps, so an agent that logs poorly produces not-determinable determinations. Remedies and compensation are outside it entirely.

Disclosure Scope

This article describes subject matter disclosed in U.S. Provisional Application No. 64/117,812 and reflects that filing as filed, including its conditioned language: where a consequence is gated on a policy-declared bound or threshold, it is stated as gated, and quantities the filing leaves to the signed policy object carry no values here.

The application is pending. Nothing here asserts that any product or service falls within any claim, and no claim scope should be inferred. References to Decagon are to public materials and are used for comparison only; no relationship, endorsement, or infringement is asserted.

Record-Grounded Conduct Admission (/conduct-admission)

All 49 steps → (/inventive-steps)

Conduct judged against the agent's own record, not an outside authority.

Provisional application

PRIMARY TECHNICAL DISCLOSURE

- Record-Grounded Conduct Admission: Governing How an Agent Answers an Accusation (/articles/conduct-admission/record-grounded-conduct-admission)

SECONDARY TECHNICAL

- Value-Scope Resolution: Testing Whether a Declared Value Is Even Implicated (/articles/conduct-admission/value-scope-resolution)
- The Retroactive Narrowing Bar: Why an Agent Cannot Edit Its Values After the Fact (/articles/conduct-admission/retroactive-narrowing-bar)
- The Delegator Floor: A Monotone Upward Ratchet on Delegated Authority (/articles/conduct-admission/delegator-floor-ratchet)
- The Co-Signed Mutual Admission Compact: Two Agents Binding Each Other (/articles/conduct-admission/co-signed-mutual-admission-compact)
- The Inherited Governance Record: Fork- and Clone-Resistant Accountability (/articles/conduct-admission/inherited-governance-record)
- The Corrective Encounter: A Commission-Versus-Omission Compliance Test for Whether an Agent Actually Changed (/articles/conduct-admission/corrective-encounter-commission-omission-compliance-test)
- Continuity-Break Demotion of a Counterparty Identity Record: Reversing First-Encounter Promotion Without a Conduct Determination (/articles/conduct-admission/demotion-persistent-tier-continuity-break)
- Ratchet-Freeze on Uncomplied Correction: Why One Conforming Act Cannot Restore an Agent's Earned Authorization Persistence (/articles/conduct-admission/ratchet-freeze-uncomplied-correction)
- Three-Way Corroboration Classification: Corroborating, Contradicting, or Neither, and the Failed-Corroboration Reject Trigger (/articles/conduct-admission/three-way-corroboration-classification-reject-trigger)
- Untested-Class Authorization Decay: Metering Agent Permission Persistence by Recorded Corrective Encounters (/articles/conduct-admission/untested-class-authorization-decay)

APPLICATIONS · GENERAL

- Agent Marketplaces: Handling a Complaint Against an Autonomous Seller (/articles/conduct-admission/agent-marketplace-dispute-handling)
- Clinical AI Agents: Answering an Accusation Without a Central Registry (/articles/conduct-admission/clinical-agent-accountability)

APPLICATIONS · SPECIFIC

- The EU AI Act and Inter-Agent Conduct: Where Record-Keeping and Oversight Duties Stop (/articles/conduct-admission/eu-ai-act-accountability-records)
- Sierra AI Agents and the Record Behind a Customer Complaint (/articles/conduct-admission/sierra-ai)
- **Decagon: Support Agents and Record-Grounded Accountability (/articles/conduct-admission/decagon-ai)**
- Intercom Fin: Resolution Rates and the Conduct Record (/articles/conduct-admission/intercom-fin)
- Zendesk AI Agents: Escalation, Audit, and Agent Conduct (/articles/conduct-admission/zendesk-ai-agents)
- Agentforce and the Accusation an Agent Must Answer (/articles/conduct-admission/salesforce-agentforce-conduct)

TERMINOLOGY

- append-only lineage field (/glossary#append-only-lineage-field)
- authorization gate (/glossary#authorization-gate)
- conduct evaluation artifact (/glossary#conduct-evaluation-artifact)
- principal (/glossary#principal)
- semantic agent (/glossary#semantic-agent)
- signed policy object (/glossary#signed-policy-object)

Record-Grounded Conduct Admission overview → (/conduct-admission)