Nick Clark · Published July 29, 2026 · PDF

Copilot Studio: Recording a Withholding as a Real Outcome

An agent that stops short of acting still produces something that has to be written down. Where the only outcome slots available are success and failure, a governed withholding lands in the failure slot and begins to accumulate as a timeout, a retry, or a mark against a connector. U.S. Provisional Application No. 64/117,812 discloses an architecture in which, in an embodiment, a withholding is a first-class outcome entry of a recorded abstention class, adverse to no party, carried in a type disjoint from magnitude by a conversion bar so that no consuming determination can turn it into a scalar, a default, a threshold operand, or a cost. This article reads that chapter alongside Microsoft Copilot Studio, an environment for building and governing custom agents.

What Happens the Moment an Agent Stops Short

A procurement agent is asked to approve a vendor change and finds the record it needs is silent on the point. It does not approve. It does not reject. What it produces has to land somewhere, and where the available outcome slots are success and failure, it lands in the second one. The caller logs a timeout, the retry policy fires, an error rate moves, and a reliability view built from those signals reports the agent as flaky.

Nothing in that chain is wrong on its own terms. Each step did what it was built to do with the value it received. The problem sits upstream: the withholding never had a representation of its own, so it arrived at each consumer as an absence, and an absence is the easiest thing in the world to coerce into a number.

The stakes rise when agents call other agents. If one agent has stopped executing a class of action and a second keeps dispatching that class to it, the second accumulates evidence of unreliability about a peer behaving exactly as its governance requires. Silence reads as fault, and recorded fault propagates on its own.

So the question this chapter takes up is narrow and structural. Not whether an agent should have declined, but what type of value the declining produces, and what a consumer of that value can do with it.

Copilot Studio in Its Own Terms

Microsoft Copilot Studio is, as publicly described, an environment for building and managing custom AI agents without writing them from scratch. Makers author conversational behavior, ground agents on organizational knowledge sources, connect them to systems of record and action through the surrounding connector ecosystem, and publish them to the channels where people already work. It sits, again as publicly described, inside a broader Microsoft platform, so identity, tenant administration, and data governance are handled where they are already handled for other workloads.

The design center, on those same materials, is authoring and operations: standing up a competent agent quickly, giving non-specialists a way to shape its behavior, and giving administrators visibility into what agents in a tenant can reach. The materials describe administrative and governance controls over agent deployment and data access, along with guardrail settings that shape agent behavior. Shortening the distance from an idea to something running under corporate policy is a hard problem, and the platform is built squarely at it.

That category differs from the one the filed architecture occupies. Agent-building platforms and guardrail layers, as a class, are organized around the decision: whether an agent should act, what it may reach, and whether the attempt is recorded. The filed architecture is organized around the value produced when the answer is neither yes nor no, and around the type discipline governing every later consumer of that value. What follows is a structural comparison, not an audit of any product.

Abstention as a Typed Outcome, Not an Empty One

In accordance with an embodiment, the filing states an invariant: no determination stage converts unavailability of a required input into an adverse consequence for any party. Where a required input is unavailable, incomplete, or unresolvable, the determination emits an outcome entry of a recorded abstention class identifying the unavailable input, and emits nothing adverse to the semantic agent (100), to an asserting party (118), or to a counterparty. The invariant applies at each stage at which an input is required, and irrespective of the cause of unavailability.

A degradation map (500) enumerates, per determination stage, the input required, the abstention outcome produced on unavailability, and the consequence foreclosed. Where the append-only lineage field (104) is unavailable, incomplete, or silent as to an asserted conduct, the admission evaluator (120) produces the not-determinable determination (126), and the consequence foreclosed is the rejected determination (124). Where the signed policy object (112) in force at the recorded assertion time cannot be resolved, no determination is produced, the conduct evaluation artifact (116) is appended as pending, and a structured inquiry requesting the policy object is generated. Where no acceptance determination arrives within the window declared in the signed policy object (112), the outcome is recorded as not-determinable, the authorization gate (300) remains in its provisional granting state, and the consequence foreclosed is resolution of the matter against the non-responding party. That window is policy-declared, with no value stated.

One path is expressly not an abstention. Where the reason-type of an edge cannot be resolved, the magnitude of modification is bounded to the non-zero minimum declared in the signed policy object (112), the consequence foreclosed being a magnitude of zero. That minimum is likewise policy-declared, no value given.

The conversion bar (502) is what makes the rest hold. It forecloses a consuming determination from converting an abstention outcome into a scalar value, into a default value, into an operand of a threshold comparison (510), or into a consequence adverse to any party. The foreclosure is affirmative rather than a gap in defined behavior: an abstention outcome is represented in a form disjoint from the domain of values a consumer can take as a magnitude, realized by a type discipline in which the outcome entry and a magnitude are values of disjoint types with no total function mapping the former to the latter. A threshold comparison of such an entry emits an outcome of the recorded abstention class and no Boolean. An accumulation over a set containing it emits an abstention outcome and not a sum over the remaining members. In an embodiment the discipline is enforced statically upon the agent's instructions before they execute, so a consuming determination attempting the conversion is not constructible.

Across the agent boundary the same discipline travels. In an embodiment, a disclosing agent that has written its authorization gate (300) to the withheld state (310) for an enumerated set of action classes, and has transitioned into the non-executing cognitive mode (302), constructs a non-execution attestation (504). Its payload carries the withheld action-class enumeration, an enumerated evidentiary basis disclosing no conduct descriptor content, an attested epoch field, an abstention-class type marker whose declared type is the abstention type (506), and an express designation that the attestation is not a determination and not a denial. A receiving agent verifies the authority credential and continuity, the successor epoch, and that the marker resolves within the closed enumeration of abstention classes in its own policy object, then writes a carried abstention entry as a value of the abstention type (506). That designation is adopted as received: not re-derived, not inferred, not defaulted.

Four conversions are then foreclosed by name. No scalar, default, or threshold operand, and no availability score, health quantity, suspicion level, or failure rate takes such an entry as input. No counter of the disclosing agent is incremented, the refusal counter (304) included. Nothing adverse is appended to that agent's counterparty identity record (114). And subsequent non-response in a withheld class is treated as neither an accepted determination (122) nor a rejected determination (124), however many such non-responses occur. The receiving agent's dispatch-authority predicate fails for the enumerated classes toward that agent alone, the failure being recorded as a positive abstention rather than a denial. Release comes only on a verified unmarked execution record, a superseding attestation omitting the class, or elapse of the time-to-live field, which elapse is expressly no evidence that the class has been restored.

Where the Two Architectures Part

Convergence sits at the level of concern: both bodies of work care about agents that must not act, and about administrators being able to see and shape that behavior.

Divergence sits in what the mechanism operates on. Governance and guardrail layers, as a class, operate on the decision and its record: what an agent may reach, and whether the attempt was captured. The filed architecture operates on the resulting value's type, and on the closure of that type under every downstream operation, including arithmetic, comparison, accumulation, and cross-agent admission. Recording an outcome does not by itself constrain what a later consumer may compute from it, and counting is where an abstention turns into a cost. The disclosed answer makes the counting operation unconstructible rather than discouraged.

A second divergence is jurisdictional. Platform governance is strongest inside its own administrative boundary, where identity, environments, and policy are managed together. The non-execution attestation (504) addresses the case outside that boundary: an agent telling an agent it does not administer that a class of action stands

withheld, in a form the receiver verifies against a continuity history and admits as a typed value without adjudicating merit. That is a peer protocol, complementary to tenant administration.

Running Both Without Overlap

The two sit at different heights of the same stack. An organization builds and governs its agents where its identity and data governance already live, the job a platform like Copilot Studio is built to do. The disclosed architecture is what an agent carries when it interoperates outward: a signed policy object declaring the closed enumeration of abstention classes, an append-only lineage field, a counterparty identity record per peer, and the type discipline over its determinations. The seam is the attestation and the carried abstention entry.

The limits of the disclosed architecture are worth stating directly. It does not decide whether an action is a good idea, and says nothing about model quality, retrieval accuracy, grounding, or the content of a response. It supplies no authoring surface, no channel integration, no tenant administration. It does not tell an operator what the emission window, the time-to-live, the non-response window, or the non-zero minimum should be, the filing declaring each of those in the signed policy object (112) without stating values. It does not prevent an agent from being wrong. It prevents a withholding from being recorded as a wrong.

Nor is it a monitoring story, those four inputs being foreclosed by name. What remains is a record of the withholding whose originally withheld action class is recoverable by following the abstention propagation chain (508) in the append-only lineage field (104), and that moves no standing quantity of either agent.

Disclosure Scope

The architecture described here is disclosed in Chapter 5 of U.S. Provisional Application No. 64/117,812, and the mechanism statements above are drawn from that document, in accordance with the embodiments disclosed there and using its own outcome words and element names. Consequences stated as following from a condition are conditional in the filing as well. Quantities the filing declares in the signed policy object (112) without stating a value are identified as such. Nothing here is a claim construction, and the application is pending.

References to Microsoft Copilot Studio are to public materials and are used for comparison only; no relationship, endorsement, or infringement is asserted.

Positive Abstention and the Conversion Bar (/positive-abstention)

All 49 steps → (/inventive-steps)

Refusing to act is a first-class, recorded outcome — and can't be laundered into acting.

Provisional application

PRIMARY TECHNICAL DISCLOSURE

- Positive Abstention: Withholding as a First-Class, Non-Convertible Outcome (/articles/positive-abstention/withholding-as-a-first-class-outcome)

SECONDARY TECHNICAL

- Relay Non-Execution Attestation With Depth and Cycle Bounds (/articles/positive-abstention/relay-non-execution-attestation)
- The Anti-Amplification Bound on Propagated Determinations (/articles/positive-abstention/anti-amplification-magnitude-bound)
- The Attestation Revocation Object (/articles/positive-abstention/attestation-revocation-object)
- Empty-Intersection Mutual Positive Abstention (/articles/positive-abstention/empty-intersection-mutual-abstention)

- Curing Counterparty Absence by Introduction Protocol (/articles/positive-abstention/introduction-protocol-cure)
- Recorded Counterfactual Coupling Test: Detecting When a Counterparty's Silence Moves an Agent's Own Authorization Gate (/articles/positive-abstention/recorded-counterfactual-coupling-test)
- Self-Execution Intersection Conjunct: Gating Cross-Agent Determination Propagation on the Admitting Agent's Own Per-Partition Execution Record (/articles/positive-abstention/propagation-admissibility-gate-self-execution-intersection)
- Attack-Conditional Cross-Boundary Suspension of Determination Propagation Between Persistent Semantic Agents (/articles/positive-abstention/propagation-admissibility-gate-attack-conditional-cross)
- Propagation Depth Ceilings for Cross-Agent Determination Records: A Monotonic Anti-Restore-Depth Gate Against Hop-Count Replenishment (/articles/positive-abstention/propagation-depth-ceiling-anti-restore-depth)
- Attestation Staleness Outcome and the Attestation Window Parameter: Bounding the Age of a Carried Withholding Without Recording a Rejection (/articles/positive-abstention/attestation-staleness-outcome-attestation-window-parameter)
- Evidentiary-Basis Omission for the Receiving Counterparty: Attesting a Withholding Without Naming the Recipient as Its Occasion (/articles/positive-abstention/evidentiary-basis-omission-receiving-counterparty)
- Persistence-Tier Emission Targeting for Non-Execution Attestations: Bounding the Audience of an Agent's Recorded Withholding (/articles/positive-abstention/non-execution-attestation-emission-targeting-persistence)
- Restoration Settlement Reference: Releasing a Carried Abstention Entry on a Matched-Pair Settlement Record of Gate Restoration (/articles/positive-abstention/restoration-settlement-reference-distinct-release-condition)

APPLICATIONS · GENERAL

- Safety-Critical Autonomy: Recording a Refusal to Act as a First-Class Outcome (/articles/positive-abstention/safety-critical-autonomy-abstention)
- Clinical Decision Support: When the Correct Output Is No Output (/articles/positive-abstention/clinical-decision-support-abstention)

APPLICATIONS · SPECIFIC

- NIST AI Risk Management Framework: Where Abstention Fits in Agent-to-Agent Conduct (/articles/positive-abstention/nist-ai-rmf-abstention)
- ServiceNow AI Agents: When the Correct Outcome Is No Action (/articles/positive-abstention/servicenow-ai-agents)

- **Copilot Studio: Recording a Withholding as a Real Outcome (/articles/positive-abstention/copilot-studio-abstention)**
- Bedrock AgentCore: Session Isolation and Governed Abstention (/articles/positive-abstention/bedrock-agentcore-abstention)

TERMINOLOGY

- abstention outcome (/glossary#abstention-outcome)
- conversion bar (/glossary#conversion-bar)
- non-execution attestation (/glossary#non-execution-attestation)
- positive abstention (/glossary#positive-abstention)

Positive Abstention and the Conversion Bar overview → (/positive-abstention)