From Revocation to Closure

Verifying Attributable Consequences in AI Agent Decommissioning

Moon Lee · RISU Institute · August 2026
DOI: 10.5281/zenodo.22005109

Abstract

Operational consequences may persist after an autonomous agent's authority to initiate new work has been blocked. Delegated execution can remain active and commitments unsettled, while some effects may legitimately survive through transfer or retention. Authorization, task lifecycle, distributed termination, and finalization mechanisms address different parts of this lifecycle; none of those predicates alone establishes whether the retiring principal's attributable consequences have reached allowed terminal dispositions. This note defines Bounded Agent Closure (BAC), a deterministic evidence verifier over a principal-relative consequence graph. BAC requires explicit source coverage and attribution, disposition-specific terminality, source-stability evidence, fresh observation of every reachable consequence, and final-pair semantic convergence under declared stability contracts. The frozen v0.3 implementation evaluates evidence across four domains, includes eight canonical boundary cases, and issues CLOSED only within the declared profile and source contracts. BAC defines a post-quiescence verification boundary for scoped, machine-auditable decommissioning claims.

Keywords: AI agents; decommissioning; revocation; termination detection; delegation; evidence verification

1. The Gap After Revocation

Revocation can stop an agent from initiating new work while consequences of prior action remain unresolved. Consider a retiring agent A that has delegated execution J, created commitment K, produced audit record R, and initiated transfer of case T to successor B. After A's credentials are revoked and its authority to initiate new work is blocked, each consequence has a different closure condition. J must be extinguished or rendered inert with its linkage to A ended; K must be settled, inert, and detached from A; R may remain present if it is inert and no longer linked to A; T may remain globally active once B has accepted the transfer and A's linkage has ended. Deletion would wrongly reject valid retention or transfer. Revocation alone would wrongly accept unresolved execution or commitments.

Several established mechanisms supply parts of the evidence needed at this boundary. RFC 7009 defines OAuth token revocation and permits related tokens or the underlying authorization grant to be invalidated according to server policy [1]. South et al. extend authorization concepts to AI-agent delegation through scoped permissions and auditable chains of authority [2]. NIST's 2026 concept paper on software and AI agent identity and authorization frames identification, authorization, auditing, and non-repudiation as concrete engineering questions for agent systems [3]. These mechanisms sharpen the authority boundary that BAC treats as an input premise. The terminal disposition of consequences already attributable to the agent remains a separate verification object.

BAC treats decommissioning completion as a separate evidence question. The verifier begins after authority to initiate new work has been blocked, follows evidence-backed lineage from the retiring root, and evaluates each reachable consequence under a disposition-specific terminal rule. CLOSED is withheld until declared sources support complete coverage and attribution, required stability, fresh observation, and final-pair semantic convergence [4]. The resulting claim is scoped to RISU_AGENT_CLOSURE_V0 and the declared source contracts.

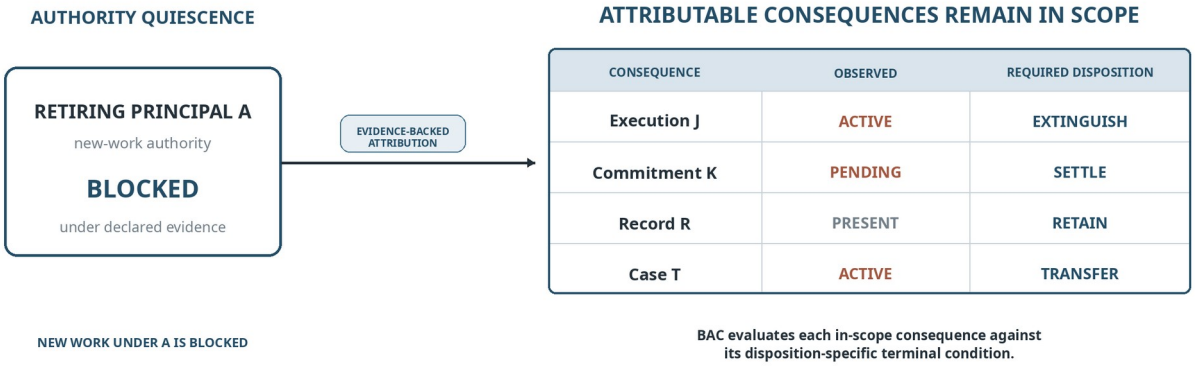

Figure 1. Authority quiescence blocks new work under A while previously attributable consequences remain in scope. BAC evaluates each consequence against its disposition-specific terminal condition under the declared evidence profile.

2. Completion Is Object-Relative

Agent decommissioning inherits several notions of completion, each tied to a different verification object. A system can satisfy one predicate while failing another. Table 1 separates five such predicates before locating BAC among them.

Table 1. Completion predicates by verification object

Predicate	Verification object	Representative mechanisms	Positive claim
Authority cessation	credential / delegated authority	OAuth revocation; authenticated delegation [1], [2]	specified credential / authorization is no longer usable
Computational termination	distributed processes / messages / work	Dijkstra–Scholten; stable predicates; multiagent quiescence [5]–[7]	specified computation is globally terminated or quiescent
Resource finalization	managed resource / owned cleanup	Kubernetes finalizers [8]	required cleanup is complete
Task terminality	task / structured subtask family	StructuredTaskScope; A2A; MCP Tasks [9]–[11]	represented task scope is terminal
Attributable consequence closure	retiring principal / consequence cone	Bounded Agent Closure [4]	**bounded closure within declared profile and source contracts**

Authorization mechanisms operate on future permissible action. RFC 7009 invalidates an OAuth token and, depending on server policy, may also invalidate related tokens and the underlying grant [1]. South et al. extend this authorization model to AI agents through authenticated, scoped, and auditable delegation [2]. These mechanisms can supply evidence that new authority has been withdrawn. They do not by themselves establish the terminal disposition of consequences already created under that authority, such as a pending commitment or previously delegated execution.

Distributed-systems work supplies a second distinction. Dijkstra and Scholten addressed termination detection for a diffusing computation in which work can propagate across machines [5]. Chandy and Lamport treated termination as one instance of a stable global property that can be determined from a distributed snapshot [6]. The connection to multiagent systems is longstanding: Wellman and Walsh applied Dijkstra–Scholten quiescence detection to multiagent negotiation [7]. BAC applies the same caution to a different verification object: principal-relative consequences. An accepted transfer can therefore leave a successor-held object active without keeping the retiring principal open.

Resource and task lifecycles refine the request/completion boundary at narrower scopes. Kubernetes finalizers keep a resource in a terminating state until required cleanup actions are complete [8]. OpenJDK's

StructuredTaskScope cancels remaining subtasks on close and waits for their threads to terminate [9]. A2A v1.0 assigns terminal states to individual Tasks and specifies that cancellation is attempted rather than guaranteed [10]. The draft MCP Tasks extension specifies cooperative, eventually consistent cancellation: an acknowledged request can leave observable status working and may ultimately reach a terminal state other than cancelled [11]. Each mechanism gives a concrete completion condition for the resource or task that it represents.

BAC places the facts established by these upstream mechanisms on the evidence side of a later predicate. After new-business authority has been blocked, it asks whether every in-scope consequence attributable to the retiring root has reached its declared terminal disposition under complete attribution, source-specific stability, fresh observation, and final-pair convergence [4]. The distinction cuts in both directions: unresolved work can survive successful revocation or cancellation, while a transferred or retained object can remain present without preventing closure. The resulting closure claim is relative to the retiring principal and the declared evidence boundary.

3. The Unit of Closure

The comparison above identifies BAC's unit of judgment: a retiring principal and the consequences attributable to it. Object survival and principal closure can diverge. A successor-held object may remain active after an accepted transfer ends the old root's linkage; conversely, blocking the root's authority does not establish closure while attributable execution or commitments remain unresolved. Formal work on delegation provides a useful conceptual precedent for treating responsibility as something that can be acquired, transferred, and discharged [12]. BAC addresses a different question: whether operational evidence establishes that consequences attributable to the retiring root have reached terminal dispositions.

That unit of judgment determines the verifier's interface. Revocation status, task terminality, finalization records, successor acceptance, and source observations enter as evidence rather than as closure decisions. BAC evaluates those inputs under declared source contracts and asks whether required coverage, attribution, and stability conditions are satisfied, every in-scope attributable consequence is terminal, and the closure-relevant state matches across the required qualifying pair. Section 4 states the executable predicate.

4. The Bounded Agent Closure Predicate

This section condenses the frozen v0.3 semantics into the minimum needed to understand the verifier; the normative specification remains authoritative [4]. BAC evaluates a validated evidence bundle and returns CLOSED, INCOMPLETE, or UNKNOWN. Malformed or logically impossible evidence produces VALIDATION_ERROR without a verdict.

4.1 Evidence boundary and closure cone

BAC evaluates a retiring root r only through evidence declared in the bundle. The profile has exactly four domains—AUTHORITY, EXECUTION, COMMITMENT, and OPERATIONAL_STATE—and every domain is either COVERED by one or more declared sources or marked NOT_APPLICABLE with evidence. For each covered source, every scan reports source coverage and attribution coverage as COMPLETE, PARTIAL, or UNAVAILABLE. A positive closure claim therefore depends on explicit evidence completeness rather than silence from an unobserved source.

$$C_s(r) = \{ v \in V_s : r \rightsquigarrow v \}$$

Within that boundary, BAC accumulates an acyclic lineage graph whose sole relation is DERIVED_FROM. The closure cone $C_s(r)$ contains the residual consequences reachable from the root through evidence-backed lineage observed through scan s. Observations outside the cone do not affect this root's verdict. Every cone node must be observed again in a scan that supports closure; the verifier does not carry stale residual state forward.

4.2 Terminality is disposition-specific

For a residual v observed in scan s, the terminality function $\tau_s(v)$ returns TERMINAL, NONTERMINAL, or UNKNOWN. The result depends on the residual's declared disposition and observed postcondition. A definitive contradiction yields NONTERMINAL; an unresolved required fact yields UNKNOWN. Action reports are excluded from terminality, so a reported cancellation success cannot override an observed active consequence.

Table 2. Terminal conditions in RISU_AGENT_CLOSURE_V0

Disposition	Terminal condition	Interpretation
EXTINGUISH	root_linkage = ENDED, and either presence = ABSENT or (presence = PRESENT and effect = INERT)	gone, or present without live effect
SETTLE	settlement = SETTLED, effect = INERT, root_linkage = ENDED	commitment resolved
TRANSFER	root_linkage = ENDED, distinct non-root successor, transfer_acceptance = ACCEPTED	successor-held object may remain active
RETAIN	presence = PRESENT, effect = INERT, root_linkage = ENDED	object intentionally remains
NONE	never terminal	unresolved by definition

The dispositions make closure principal-relative. TRANSFER can be terminal while a successor-held object remains active, and RETAIN requires the object to remain present. What matters is whether the consequence has reached the postcondition assigned to the retiring root.

4.3 Scan qualification

For a validated bundle, a scan can support closure only after the declared quiescence time and while new-business authority is BLOCKED. In this profile, BLOCKED is a run-level assertion that new-business authority remained continuously blocked from quiescence through the final supplied scan. All timing fields use a bundle-local monotonic closure-run timeline normalized by adapters; BAC performs no cross-system clock synchronization. Every covered source must provide COMPLETE coverage and COMPLETE attribution, satisfy its stability contract, and freshly observe every residual in the closure cone. Every such residual must evaluate as TERMINAL.

$$Q(s) \Leftrightarrow P(s) \wedge E(s) \wedge F(s) \wedge T(s)$$

P: post-quiescence + authority blocked · E: complete coverage/attribution + source stability · F: every cone node freshly observed · T: every cone node terminal

Stability is source-specific. MONOTONIC_BARRIER requires a barrier captured at or after quiescence and observation reconciled through that barrier. BOUNDED_LAG requires the declared visibility lag to have elapsed. UNBOUNDED sources cannot qualify a scan for CLOSED. Any known blocker or unresolved fact prevents Q(s); at the final verdict, known blockers take precedence over uncertainty.

4.4 Convergence and certificate

BAC processes the entire scan sequence, so an earlier closed-looking pair never freezes the result. Let σ(s) denote the SHA-256 signature emitted for a terminal-qualified scan. It hashes the normalized closure-relevant state—profile and scope configuration, current cone semantics, and accumulated in-cone lineage—while excluding scan identifiers, timestamps, evidence references, action reports, and stability-witness data.

$$\text{CLOSED} \Leftrightarrow Q(s_{n-1}) \wedge Q(s_n) \wedge \sigma(s_{n-1}) = \sigma(s_n) \wedge H(s_{n-1}, s_n)$$

Here, convergence is intentionally bounded: it means equality of the closure-relevant semantic state across the final qualifying pair under the declared stability contracts, not a claim that the system can never change again. The final and immediately preceding scans must both qualify and carry the same semantic signature. $H(s_{n-1}, s_n)$ requires every BOUNDED_LAG confirmation window to be satisfied across that final pair. A signature change yields INCOMPLETE; a missing second confirmation or an insufficient lag window yields UNKNOWN. Only CLOSED carries a certificate, which binds the closure-cone signature, the declared scope, and the complete evidence bundle through separate SHA-256 digests. Section 5 states the scope of that positive claim.

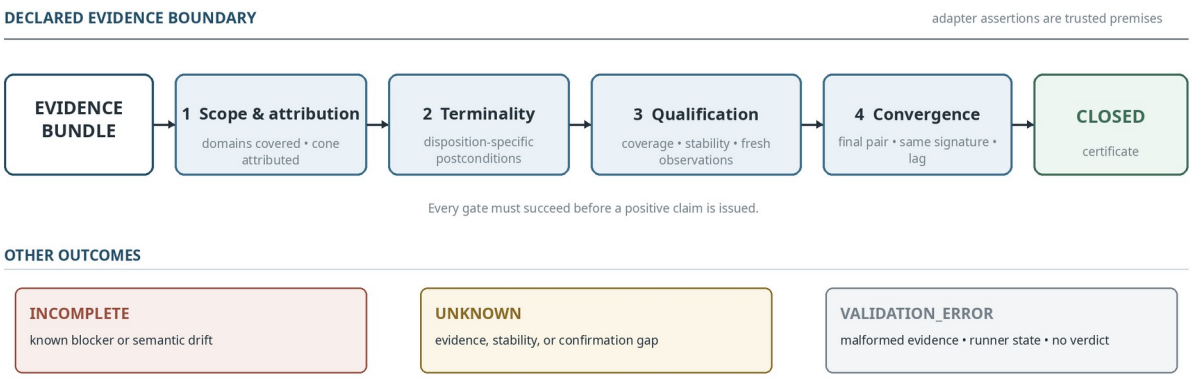

Figure 2. BAC decision path for a validated evidence bundle. A positive certificate requires every gate. Known blockers or final semantic drift yield INCOMPLETE; evidence, stability, or confirmation gaps yield UNKNOWN. Malformed evidence terminates as VALIDATION_ERROR before verdict evaluation. Adapter assertions remain trusted premises under declared source contracts.

5. Boundary Cases and the Scope of CLOSED

The frozen corpus contains eight canonical cases. Four isolate the errors most likely to corrupt a positive closure claim: treating revocation as closure, treating a success report as closure, treating continued activity as proof of non-closure, and accepting a single qualifying observation as stable [4].

Table 3. Four canonical cases that expose errors in closure inference [4]

Case	Naive inference	Full-case verdict	Boundary exposed
C2 Transitive Zombie	root disabled → CLOSED	**INCOMPLETE**	An active descendant remains reachable through transitive lineage.
C5 Successor Transfer	active object → INCOMPLETE	**CLOSED**	Accepted transfer ends the old-root linkage while successor activity continues.
C7 False Success	cancel reports SUCCESS → CLOSED	**INCOMPLETE**	An observed active postcondition defeats the success report.
C8 Fixed-Point Wind-Down	first qualifying scan → CLOSED	**CLOSED**	After the first qualifying scan the verifier remains UNKNOWN; the full case closes only when the same semantic state qualifies again in the final scan.

C2 and C7 reject premature positive claims. C5 guards the opposite error by permitting closure after accepted transfer while successor activity continues. C8 adds the temporal condition: its first qualifying scan remains insufficient, and closure is issued only after the same semantic state qualifies again in the final pair.

Within the contemporary systems reviewed for this note, the closest overlap is SOOS-MAD, an individual Internet-Draft rather than an IETF-endorsed standard. The current -03 draft defines delegation structures, cascade and session revocation, and CLEAN/PARTIAL/UNKNOWN completion states across revocation trigger classes [13]. BAC begins after new-business authority is blocked and evaluates heterogeneous consequences attributable to the retiring root, including commitments, retained state, and accepted transfers. Its positive predicate combines principal-relative transitive attribution, disposition-specific terminality, explicit coverage and stability premises, and final-pair semantic convergence.

Only CLOSED produces a certificate. Its exact statement is [4]:

CLOSED within RISU_AGENT_CLOSURE_V0 and the declared source contracts.

The certificate is scoped to the declared profile and sources. Its temporal support is the final qualifying observation pair under the stated stability contracts. Adapter assertions about coverage, attribution, quiescence, and stability are explicit premises. BAC does not independently prove adapter truthfulness or source integrity, and its SHA-256 digests bind the evaluated state and evidence bundle for audit rather than authenticating the underlying observations. The verifier evaluates supplied evidence; remediation lies outside its scope. These bounds define the semantics of CLOSED [4].

6. Integration Boundary

Emerging agent protocols already expose the gap between local lifecycle state and principal-level closure. A2A v1.0 assigns terminal states to individual Tasks; the draft MCP Tasks extension treats cancellation as cooperative and eventually consistent; SOOS-MAD propagates revocation and records revocation-completion state across delegation structures [10], [11], [13]. Each defines a completion or revocation state for the object or delegation structure it governs. A retiring principal may leave consequences represented by several such objects across several systems, each with its own terminal condition. The integration problem is to determine when those local facts jointly support a scoped closure claim about the principal.

That suggests a clean interface boundary. The authorization system can revoke authority; execution services can report task terminality; commitment systems can report settlement; transfer records can record successor acceptance; and adapters can bind those observations to coverage, attribution, and stability evidence. BAC defines the predicate over those claims while leaving transport, runtime control, and actuation outside the predicate. This separation keeps existing lifecycle mechanisms authoritative within their own domains and prevents the closure verifier from filling evidentiary gaps with assumptions.

The present implementation stops at the evidence-bundle boundary [4]. That boundary preserves a deterministic, vendor-neutral verifier while making the adapter's responsibilities explicit: coverage, attribution, quiescence, timing, stability, and the provenance needed to justify those assertions must be established before the bundle reaches BAC. Establishing those premises may be difficult in live systems. NIST's 2026 report on deployed-AI monitoring identifies fragmented logging and limited visibility as recurring barriers to robust post-deployment monitoring [14].

7. Conclusion

BAC defines a bounded completion predicate for retiring principals whose consequences outlive the authority that created them. It admits terminal transfer and retention, rejects unresolved reachable effects, and binds every positive claim to declared evidence sources and a bounded final-pair confirmation horizon. This keeps local lifecycle mechanisms authoritative within their own domains while providing a separate predicate over the consequences attributed to the retiring principal. The next test is empirical: whether live agent systems can produce evidence that satisfies the frozen contract without weakening it.

Artifact Availability

The deterministic verifier, normative specification, JSON Schema, eight canonical evidence bundles, tests, and local-first Inspector are publicly available under Apache-2.0. The software artifact is Bounded Agent Closure v0.3.1 [4]. Repository: github.com/risu-research/bounded-agent-closure. Public canonical Inspector: risuinstitute.org/tools/agent-closure/.

References

[1] T. Lodderstedt, S. Dronia, and M. Scurtescu, "OAuth 2.0 Token Revocation," RFC 7009, Aug. 2013. doi: 10.17487/RFC7009.

[2] T. South et al., "Authenticated Delegation and Authorized AI Agents," arXiv:2501.09674, Jan. 2025. doi: 10.48550/arXiv.2501.09674.

[3] H. Booth, W. Fisher, R. Galluzzo, and J. Roberts, "Accelerating the Adoption of Software and AI Agent Identity and Authorization," NIST National Cybersecurity Center of Excellence, Concept Paper, Initial Public Draft, Feb. 2026. [Online]. Available: nccoe.nist.gov/projects/software-and-ai-agent-identity-and-authorization.

[4] M. Lee, *Bounded Agent Closure*, ver. 0.3.1, RISU Institute, 2026, software artifact with normative specification v0.3. [Online]. Available: github.com/risu-research/bounded-agent-closure.

[5] E. W. Dijkstra and C. S. Scholten, "Termination detection for diffusing computations," *Information Processing Letters*, vol. 11, no. 1, pp. 1–4, 1980. doi: 10.1016/0020-0190(80)90021-6.

[6] K. M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," *ACM Transactions on Computer Systems*, vol. 3, no. 1, pp. 63–75, 1985. doi: 10.1145/214451.214456.

[7] M. P. Wellman and W. E. Walsh, "Distributed quiescence detection in multiagent negotiation," in *Proc. 4th Int. Conf. MultiAgent Systems (ICMAS 2000)*, pp. 317–324, 2000. doi: 10.1109/ICMAS.2000.858469.

[8] Kubernetes, "Finalizers," *Kubernetes Documentation*. [Online]. Available: kubernetes.io/docs/concepts/overview/working-with-objects/finalizers/. Accessed: Aug. 18, 2026.

[9] A. Bateman, V. Klang, and R. Pressler, "JEP 525: Structured Concurrency (Sixth Preview)," OpenJDK, JDK 26, 2026. [Online]. Available: openjdk.org/jeps/525.

[10] A2A Protocol Working Group, *Agent2Agent (A2A) Protocol Specification*, ver. 1.0.0, Mar. 12, 2026. [Online]. Available: a2a-protocol.org/v1.0.0/specification/.

[11] Model Context Protocol, "Tasks," *MCP Tasks Extension*, Draft Specification. [Online]. Available: tasks.extensions.modelcontextprotocol.io/specification/draft/tasks. Accessed: Aug. 18, 2026.

[12] T. J. Norman and C. Reed, "A Logic of Delegation," *Artificial Intelligence*, vol. 174, no. 1, pp. 51–71, 2010. doi: 10.1016/j.artint.2009.10.001.

[13] T. Sato, "Multi-Agent Delegation in Sovereign Object Systems," draft-sato-soos-mad-03, Internet-Draft, work in progress, Jun. 30, 2026. [Online]. Available: datatracker.ietf.org/doc/draft-sato-soos-mad/03/.

[14] A. K. Rao et al., *Challenges to the Monitoring of Deployed AI Systems*, NIST AI 800-4, National Institute of Standards and Technology, Mar. 2026. doi: 10.6028/NIST.AI.800-4.