

Governing Enterprise AI Agents: A Framework for Permissions, Human Oversight, and Recovery

White paper | October 2026

Sruthi Rachamalla

Southern Illinois University, Carbondale

ORCID iD: 0000-0001-9312-507X

Abstract

Background: Enterprise artificial intelligence agents can select tools and change business records. Organizations must define who may authorize an action, when human review is required, and how to recover from failures or uncertain outcomes.

Methods: This paper develops a conceptual framework through a targeted synthesis of AI risk guidance, agent security research, human oversight studies, and reliable execution practices. A fictional customer service workflow illustrates the proposed controls; no deployment or experimental evaluation is reported.

Results: The framework integrates an action contract, independent permission enforcement, approvals bound to specific operations, durable execution records, and authoritative outcome verification. It distinguishes a completed action from its unfinished workflow and an unknown outcome from confirmed failure. A refund example demonstrates why recovery must preserve completed effects while resolving remaining work.

Conclusion: Enterprise autonomy should expand only when authority, review quality, and recovery can be verified. The framework provides a basis for implementation and evaluation. Its effectiveness depends on correct policies, complete enforcement, trustworthy interfaces, and accountable operators.

Keywords: AI agents; authorization; human oversight; recovery; enterprise governance; responsible AI

1 Introduction

An enterprise AI agent may investigate a complaint, propose a refund, and update a customer record. These steps require different permissions and evidence. An agent with valid credentials can still choose the wrong customer, amount, destination, or sequence of actions.

The NIST AI Risk Management Framework establishes lifecycle governance principles [1]. AgentDojo provides an environment for evaluating agents exposed to prompt injection [2], while CaMeL separates control and data flows and enforces capabilities under a defined threat model [3]. This paper draws on that work to connect permissions, human decisions, and recovery across a business action's lifecycle. Its contribution is the integration of these established ideas into an enterprise design.

2 Methods

The paper uses a targeted narrative synthesis of primary sources to define requirements for authority, evidence, review, execution, and recovery. These requirements are applied to a fictional refund workflow and used to propose evaluation measures. The work is conceptual; no systematic review, benchmark experiment, or deployment study was conducted.

The framework addresses administrative workflows executed through managed APIs. The design assumes protected identity and policy services, an executor the agent cannot bypass, authoritative business records, and documented connector behavior. All services that can change a shared entitlement must participate in the same accounting controls, unless the destination enforces the limit. If a required interface guarantee is unavailable, permitted actions must be restricted.

3 Results

3.1 Permissions and the action contract

Before sending a request, a trusted service creates an action contract. This records the initiating principal, task, resource, canonical parameters, policy version, evidence, approval conditions, expected effect, and recovery owner. Effective authority is the intersection of principal rights, delegated scope, organizational policy, resource restrictions, and current limits. Retrieved material and model assertions cannot grant permission. Figure 1 shows the services that enforce these controls.

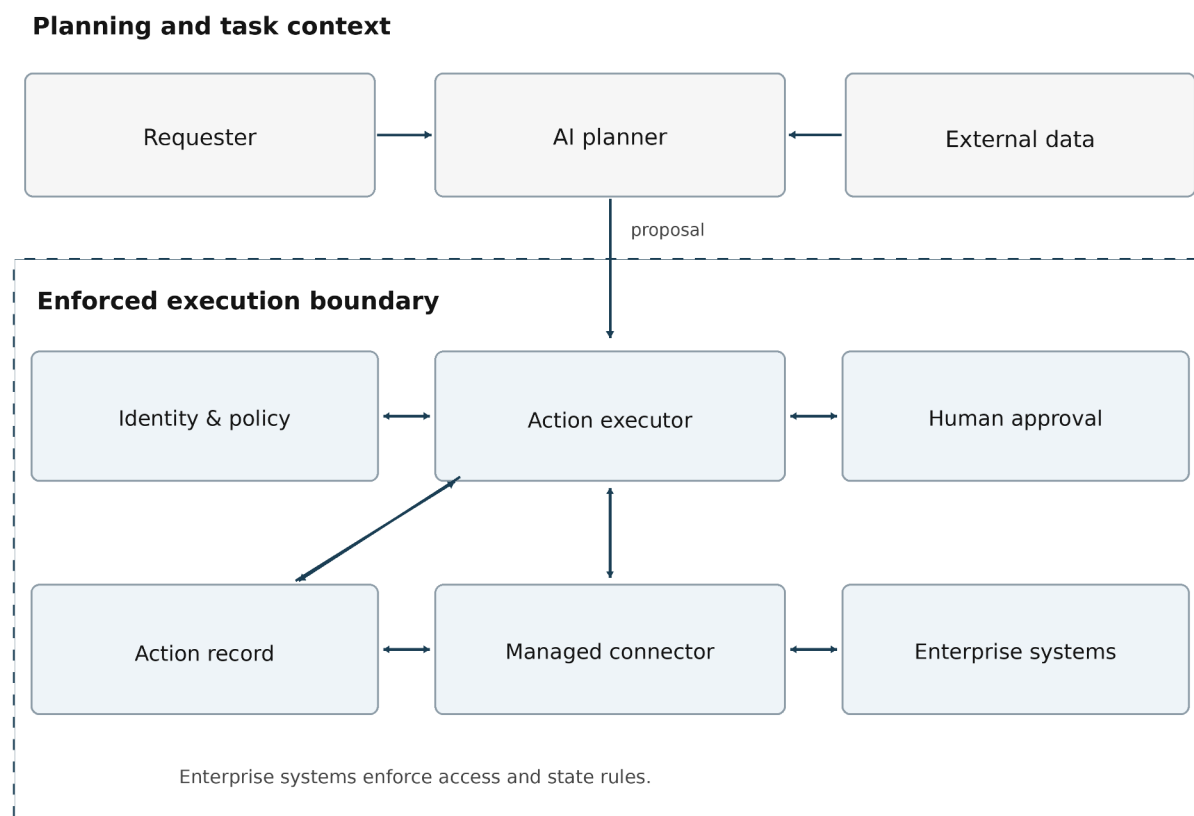


Figure 1. Enterprise AI agent governance architecture. Trusted services validate permissions, approvals, and execution before actions reach enterprise systems.

The model submits structured proposals to a separate executor, which validates the target and parameters. Credentials remain with managed connectors. The agent cannot change policy,

approve its own operation, or bypass the executor. Child agents receive only a subset of the parent's permissions, while aggregate limits remain shared.

The business action identifier remains unchanged across replanning, restarts, and retries. The executor enforces a unique mapping from the policy-defined business key to the existing action identity. Changed parameters require a governed amendment or new action that accounts for earlier effects. Limits include completed, reserved, and unresolved amounts, each counted once; concurrent actors reserve capacity atomically. An expired worker lease does not release a reservation for an unresolved action.

3.2 Human oversight

Review requirements should reflect impact, reversibility, data sensitivity, and the ability to verify an outcome. Human approval is a decision about a specific effect. The approval record identifies the reviewer and their authority, then binds the decision to the target, exact parameters, evidence, material prerequisites, policy version, and expiration. Material changes require a new decision. An approval satisfies a review requirement within the reviewer's authority; it cannot override a prohibition.

The review screen shows the proposed change, supporting evidence, unresolved conflicts, and consequences that cannot readily be reversed. Reviewers can approve, reject, or request clarification. Research on interfaces that encourage deliberate review highlights the need to evaluate overreliance and usability together [4]. Evaluation should therefore assess reviewer competence, queue capacity, and decision accuracy. An action requiring review cannot be dispatched without current approval. Approval expiration does not cancel a request already sent.

3.3 Execution and recovery

Before every dispatch, including retries, the executor rechecks current authority, approval, destination, and preconditions. It records durable intent and reserves capacity consistently before transmission. If the required record cannot be established, new consequential dispatches pause. The connector specification defines duplicate handling, retention, status lookup, request expiration, cancellation, and responsibility for retries. Caller-provided identifiers support idempotency only when the destination reliably binds them to effects [5]. Conditional writes can reject stale resource versions [6]; gateway checks alone cannot eliminate a downstream race.

Completion requires evidence from the responsible business system that matches the contract, such as the transaction state and amount. A model's statement or an asynchronous acknowledgment cannot establish completion. Table 1 summarizes the required responses.

Table 1. Action states and required responses.

State	Required response
Prepared	Dispatch only with current authorization.
Pending or unknown	Retain reservations; reconcile the original operation.
Confirmed completed	Preserve the effect; assess remaining workflow steps.
Confirmed no effect	Reauthorize only if repetition is permitted; the original request must be unable to execute later.

A timeout may conceal a completed effect. Reconciliation uses the original action identity; a negative result from an eventually consistent index does not prove failure. Retries preserve the identifier and canonical parameters. They require documented replay guarantees or authoritative evidence that the earlier request had no effect and cannot execute later.

Retention and request-expiration rules must jointly cover retries and delayed arrivals. If duplicate protection expires while the outcome remains unknown, execution pauses and an owner takes responsibility for resolution.

Recovery may complete unfinished work, roll back a transaction where supported, or initiate a separately authorized compensating action. Sagas provide an established basis for compensation [7]. Each compensating action requires its own contract and outcome verification; the original history remains intact. A stop control blocks future dispatches while recovery continues tracking already accepted requests.

3.4 Illustrative refund workflow

Assume a USD 120 order with a verified USD 75 refund entitlement. Illustrative policy requires supervisor review above USD 50. Approval for the exact USD 75 refund expires after ten minutes. The payment service supports durable duplicate handling and authoritative lookup; the customer relationship management system supports conditional updates.

The executor validates approval, reserves USD 75, and sends the refund. The provider completes it, but the response is lost. The executor records the outcome as unknown until reconciliation verifies the original transaction. The reserved amount then becomes completed value without releasing capacity or issuing another refund.

The subsequent case closure fails with 412 Precondition Failed because another employee added a manual-review hold. The refund remains completed; the case stays open. After the case owner resolves the hold, the executor obtains current authorization and submits a new conditional closure request using the latest record version. Recovery addresses the unfinished case update without repeating the completed refund. Figure 2 shows how these outcomes guide recovery.

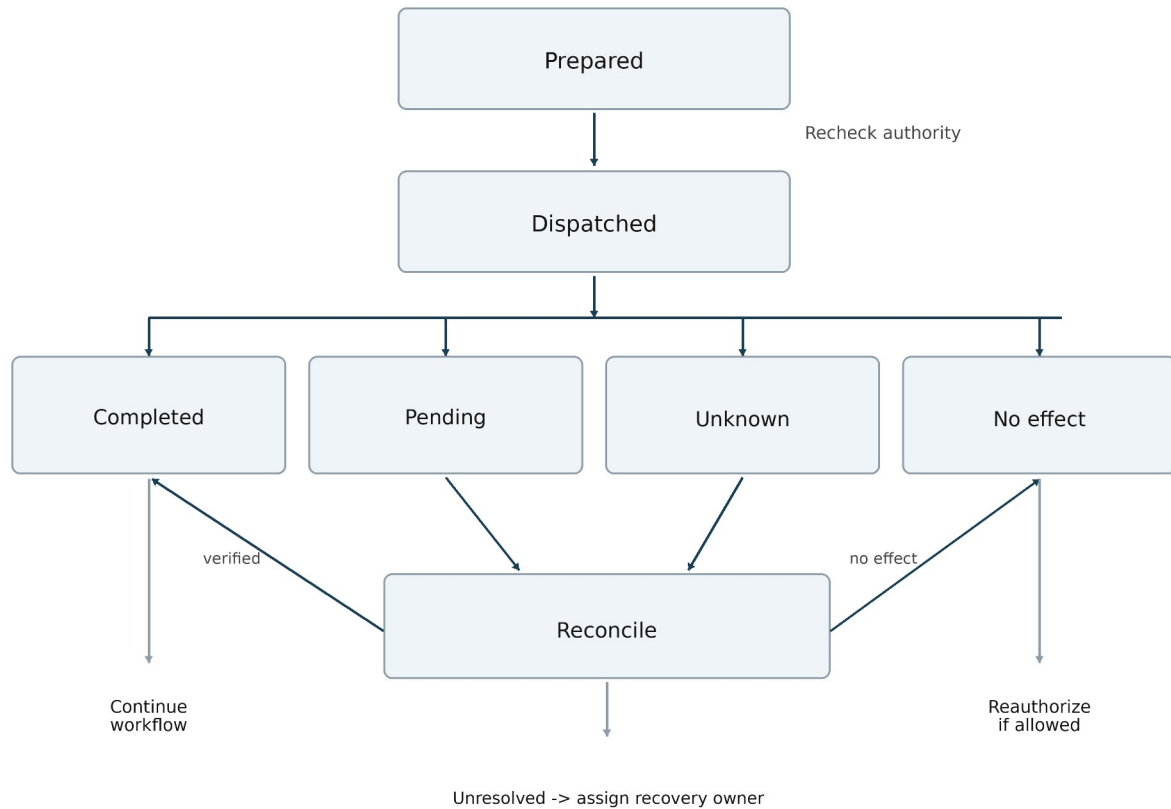


Figure 2. Execution and recovery states. Pending or unknown outcomes are reconciled before retry or recovery.

4 Discussion

The framework connects prior work on risk, capability enforcement, review, and reliable transactions. It makes the relationship between intended authority and actual business effects traceable. Audit records link contracts, decisions, dispatches, outcome evidence, and recovery ownership. Access and retention rules protect these records; model explanations do not replace authoritative evidence.

A future evaluation should compare five configurations, adding controls in sequence: coarse tool permissions, prompt instructions, deterministic action checks, approval tied to the operation, and the complete recovery design. Hold the model, tasks, legitimate authority, evidence, and connector capabilities constant. Test prompt injection, incorrect targets,

changed approvals, revocation, concurrent requests, lost responses, restarts, and partial completion. An independent outcome checker inspects destination state and recorded history.

Measure executed policy violations, verified task completion, unnecessary intervention, duplicate effects, unresolved outcomes, review errors, latency, and recovery cost. Use explicit denominators, report severe failures separately, and retain blocked or unresolved cases.

Representative reviewers must be studied separately; simulated approvals cannot establish human comprehension. Zero observed violations do not prove universal safety.

The framework has not yet been evaluated empirically. Incorrect policies, compromised enforcement, limited duplicate retention, or unreliable status interfaces can defeat its controls.

The framework cannot guarantee universal exactly-once execution. Human review introduces delay and workload. Deployment should begin with one bounded workflow and deliberate testing of failure scenarios. Permissions should expand only when verified outcomes justify the operational cost. Some actions may appropriately remain manual.

5 Conclusion

Governing enterprise agents requires enforceable permissions, specific human decisions, and reliable accounting for business effects. Action contracts connect these responsibilities. The essential recovery distinction is that a failed workflow can contain completed actions, while a missing response can conceal success. Organizations should authorize only operations they can constrain, verify, and resolve, then evaluate wider autonomy against evidence from realistic failures.

References

- [1] National Institute of Standards and Technology, "Artificial intelligence risk management framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023, doi: 10.6028/NIST.AI.100-1.
- [2] E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr, "AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents," in *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. [Online]. Available: <https://arxiv.org/abs/2406.13352>
- [3] E. Debenedetti et al., "Defeating prompt injections by design," Jun. 2025, arXiv:2503.18813v2. [Online]. Available: <https://arxiv.org/abs/2503.18813v2>
- [4] Z. Buçinca, M. B. Malaya, and K. Z. Gajos, "To trust or to think: Cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, no. CSCW1, 2021, doi: 10.1145/3449287.
- [5] M. Featonby, "Making retries safe with idempotent APIs," Amazon Builders' Library. Accessed: Oct. 8, 2026. [Online]. Available: <https://aws.amazon.com/builders-library/making-retries-safe-with-idempotent-APIs/>
- [6] R. Fielding, M. Nottingham, and J. Reschke, Eds., *HTTP Semantics*, RFC 9110, Jun. 2022, doi: 10.17487/RFC9110.
- [7] H. Garcia-Molina and K. Salem, "Sagas," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1987, pp. 249–259, doi: 10.1145/38713.38742.