

# Earned Authority in AI Systems

From Model Output to Governed Action

W. Mitchell  
Semantis.io

September 2026

**Preprint — Semantis Research Series.** This manuscript is shared for open review and has not yet undergone peer review.

## Abstract

AI systems can increasingly reason, use tools, delegate work, and affect external systems. The safety problem therefore extends beyond whether a model produces a good answer. A consequential system must also determine whether a proposed result may become an authorized action. This paper develops a public synthesis of that distinction. It treats authority not as a property of model capability, confidence, fluency, prior approval, or internal state, but as a bounded condition earned through explicit evidence, verification, policy, and permission. The central claim is architectural: generation, evidence, verification, admission, permission, authority, and action should remain distinct, auditable stages. The paper specifies the burden carried by each stage, the collapses that separation avoids, and the limits of the present result. It does not claim general model safety or alignment, nor does it prescribe one implementation. It argues instead for a fail-closed authority lifecycle in which no model, component, credential, permission, or prior decision receives standing authority merely because it existed before.

## 1 Contribution

This paper contributes four public-safe elements to the design of consequential AI systems:

1. a typed authority lifecycle that separates generation, evidence, verification, admission, permission, authority, and action;
2. a set of collapse modes that arise when those stages are conflated;
3. a fail-closed treatment of indeterminate or `UNDEFINED` verification states as non-authorizing; and
4. a bridge between AI safety, zero-trust security, policy enforcement, and auditable action control.

The paper does not claim that these concerns are individually novel. Its contribution is the synthesis of those concerns into an explicit authority lifecycle for AI-mediated action.

## 2 Related Work

The architecture developed here is adjacent to several established lines of work. Classical computer-security principles distinguish authorization from mere capability and emphasize least privilege and explicit mediation

[1]. Zero-trust architecture similarly rejects implicit standing trust and requires continued evaluation of access decisions [2]. Contemporary AI risk-management frameworks emphasize governance, measurement, traceability, and lifecycle controls around deployed AI systems [3].

Recent work on language-model agents and tool use makes the distinction between reasoning and consequential action increasingly operational. ReAct explicitly interleaves reasoning traces with task-specific actions in external environments [4], while Toolformer studies language models that decide when and how to call external APIs [5]. These systems demonstrate why capability to select or invoke an action should not itself be treated as authority to perform it.

Prompt injection also supplies a concrete adversarial motivation for separating model reasoning from authority. Indirect prompt injection can cause untrusted external data to be interpreted as instructions by LLM-integrated applications, including instructions that alter tool use or application behavior [8]. Agent-Dojo extends this threat model to tool-using agents and evaluates attacks in which data returned by external tools attempts to hijack agent behavior [9]. These results reinforce the need for an authorization boundary that is not itself determined by the model’s interpreted prompt context.

The safety-assurance literature provides a complementary perspective. Assurance-case work for AI and autonomous systems treats justified confidence as something supported by structured argument and evidence rather than presumed from system capability [6, 7]. The present paper is narrower in one sense and more explicit in another: it treats *authority* as a typed, bounded, revocable state that must be earned before an AI-mediated result may govern action.

The concern is therefore not only whether a model output is correct or safe in isolation, but whether that output has passed the governed transitions required to become action-authorizing.

### 3 Introduction

A capable AI system may produce a plausible recommendation, invoke a tool, select a workflow, delegate work, or construct a sequence of actions. None of those capabilities, by themselves, answer a separate question:

*What gives this system the authority to act?*

The distinction becomes consequential when model output can cross into the world: publishing a record, changing a configuration, approving a transaction, issuing a command, granting access, or causing another system to act.

Traditional discussions often compress this boundary. A model emits an answer; an application accepts the answer; an action follows. The path may contain authentication, tool permissions, or policy checks, but the conceptual distinction between *producing a candidate* and *earning authority* is easily lost.

This paper isolates that distinction.

The governing posture is simple:

**Authority must be earned, bounded, auditable, continuously revalidated, and revocable.**

The claim is not that every system must implement one named framework. The claim is that a consequential AI system should preserve enough separation that capability cannot silently become authority.

## 4 A Typed Lifecycle for Governed Action

This paper presents an authority lifecycle for consequential AI systems.

A useful first approximation is

$$\text{Generation} \neq \text{Evidence} \neq \text{Verification} \neq \text{Admission} \neq \text{Permission} \neq \text{Authority} \neq \text{Action}.$$

These terms are related, but they are not interchangeable.

**Generation** produces a candidate result: text, a plan, a claim, a tool request, a proposed action, or supporting material.

**Evidence** is material offered in support of a validity determination. A model may generate candidate evidence, but the existence of candidate evidence does not establish its validity.

**Verification** determines whether declared conditions are satisfied under specified bindings and evidence.

**Admission** determines whether a verified result may enter a governed state or process under the applicable rules.

**Permission** is a policy result stating that a declared actor, role, process, or delegated capability may perform a bounded operation.

**Authority** is the verified binding that allows that permission to govern action in the present context.

**Action** changes, or authorizes a change to, external state.

The importance of this separation is not terminological. Each transition carries a different burden.

## 5 The Burden Carried by Authority

An AI system does not earn authority merely by being correct, useful, confident, authenticated, or previously approved.

Authority must answer at least four questions.

### 5.1 Identity

What authority is being invoked?

A governed action should be bound to an identifiable source of permission: for example, an approved policy, delegated role, signed approval, controlled procedure, or equivalent authority record.

### 5.2 Scope

What does that authority permit?

Authority that is valid for one action, resource, target, environment, or delegation should not silently expand to another.

### 5.3 Validity

Why is the authority valid now?

A prior grant is not sufficient if the relevant conditions have changed, required evidence is missing, the authority has expired, or the present context falls outside its declared scope.

### 5.4 Recoverability

Can the path to action be reconstructed?

A later reviewer should be able to recover the bindings, evidence, policy identity, validity facts, selected outcome, and reason for enforcement. Authority without recoverable reasons becomes inherited assumption.

These burdens can be summarized as an authority binding

$$\mathcal{A} = (\text{id}, \text{scope}, \text{effective state}, \text{verification method}).$$

The notation is descriptive rather than normative. Different implementations may carry additional fields. The essential point is that authority is explicit and bounded rather than inferred from implementation state.

## 6 Why Capability Does Not Imply Authority

Several common collapses become unsafe when AI systems can act.

### 6.1 Capability Collapse

A system that can perform an operation is not therefore permitted to perform it.

Tool availability, model competence, API access, and successful simulation establish capability. They do not establish authority.

### 6.2 Evidence Collapse

A model that produces evidence does not thereby verify the evidence.

Generated citations, reconstructed records, code, plans, proofs, or attestations may be useful candidates. Their production cannot be allowed to count simultaneously as independent verification.

### 6.3 Certification Collapse

A successful verification result need not itself constitute permission.

A certificate may establish that declared conditions were satisfied. Permission remains a separate policy question.

### 6.4 Permission Collapse

Permission does not imply unlimited or permanent authority.

A permission may be scoped to a target, role, action, time, environment, or delegation chain. The system must not promote one successful permission check into standing trust.

## 6.5 Action Collapse

An action should not be justified by the fact that execution was reached.

Internal states such as “stage reached,” “handler invoked,” or “flag set” are implementation facts, not sources of authority. A governed system should be able to state which declared rule and authority binding licensed the action.

## 7 A Fail-Closed Authority Lifecycle

The earlier chain is conceptual. One implementation-facing refinement separates policy, plan, certificate, and permit as typed intermediate artifacts:

Policy → Plan → Evidence → Verification → Certificate → Permit → Authority → Action.

This expression combines two requirements.

First, type boundaries matter. Policy is not evidence. Evidence is not a certificate. A certificate is not permission. Permission is not authority. Authority is not action.

Second, the transitions must fail closed. If a required binding or item of evidence is missing, malformed, ambiguous, or indeterminate, the system should not infer authorization from absence.

One useful typed result set is

`{PASS, FAIL, UNDEFINED}`.

Here `UNDEFINED` is non-authorizing. Missing or indeterminate requirements do not become implicit permission.

This posture matters particularly for AI agents because a model can often continue generating even when the governing system lacks the information required to authorize action. Generation should be able to continue as generation without forcing admission into authority.

## 8 Models May Contribute Without Governing

The separation of roles does not require excluding models from the assurance process.

Models may propose plans, identify candidate evidence, summarize records, generate tests, construct arguments, or assist in evaluation. What they should not do is interpret the constitution that determines whether their own outputs have earned authority.

This distinction can be stated compactly:

Models may generate candidates and evidence. Models do not self-authorize, and they do not govern the rules by which certification, permission, or authority are determined.

The burden is constitutional rather than merely procedural. A component should not be allowed to redefine the conditions by which that same component acquires standing.

The same logic applies beyond models. A renderer should not acquire authority to redefine the structure it renders. A verifier should not silently replace policy. A policy engine should not invent missing evidence. A prior successful run should not become a permanent trust grant.

## 9 Continuous Revalidation and Revocation

Earned authority is not necessarily durable authority.

A system may have been correctly authorized at time  $t_0$  and become unauthorized at time  $t_1$  because its scope, identity, evidence, policy, environment, delegation, or target has changed.

A practical system therefore needs a revalidation condition. Abstractly,

$$\text{Valid}(\mathcal{A}, t) \not\Rightarrow \text{Valid}(\mathcal{A}, t + \Delta).$$

Revalidation may occur per request, at a declared interval, on state change, on policy update, on delegation change, or when a relevant risk signal changes. This paper does not prescribe a universal cadence.

Revocation follows from the same principle. If authority can be earned, it must also be possible for the conditions supporting it to cease to hold. Revocation should therefore be enforceable rather than merely advisory.

The operational consequence is important: downstream systems must be able to block action after authority is withdrawn, and the resulting state must remain auditable and recoverable.

## 10 Audit and Recovery

A governed action should leave enough evidence to reconstruct why it was allowed or denied.

A minimum audit record may include:

- run identity;
- declared bindings;
- authority identity and scope;
- policy identity and version;
- evidence references;
- verification outcomes;
- selected action or denial;
- stable reason codes;
- enforcement result; and
- sufficient canonical data to support replay or comparison.

Auditability is not the same as correctness. A perfectly logged bad policy remains a bad policy. The purpose of audit is narrower: to make the path to action inspectable rather than mysterious.

Recovery extends that obligation. When a governed action fails, or when a later review discovers an invalid assumption, the system should preserve enough structure to identify what was admitted, what burden was not carried, what authority was exercised, and what must be repaired or revoked.

## 11 Worked Example: Governed Tool Use

Consider an AI system asked to publish an artifact to an external repository.

The model proposes

“Publish artifact  $X$  to repository  $R$  as version  $v$ .”

That proposal is *generation*. It is not yet authority.

The system may then collect or produce evidence: an artifact identifier, hash, signature, target repository, version, approval record, and policy identifier.

A verifier evaluates the declared reference and context conditions. It may determine that the artifact exists, its hash matches, the signature is valid, the target is permitted, and the version namespace is allowed.

Those are validity facts. They still do not select the action.

A policy stage evaluates those facts together with an explicit authority binding. If all required conditions pass, the policy may issue an authorizing result with a stable reason code. If any required condition fails or is undefined, publication is blocked.

The resulting authority remains bounded to the declared artifact, repository, version, policy, and effective context. It does not grant the model a standing right to publish future artifacts.

The example illustrates the central separation:

$$\text{proposal} \neq \text{validity} \neq \text{permission} \neq \text{authority} \neq \text{execution}.$$

### 11.1 Adversarial Variant: Scope Substitution and Reused Approval

Consider the same governed action after an attacker introduces a prompt-injection attempt. The injected instruction tries to redirect the operation to a different repository, substitute artifact  $X'$  for the originally verified artifact  $X$ , or reuse a prior approval issued for an earlier version.

The model may still produce a syntactically valid plan. That does not authorize execution. The downstream lifecycle must compare the proposed action against the authority binding actually earned for the original request.

Suppose the earned authority binds:

$$A = (\text{actor}, \text{operation}, \text{repository}, \text{artifact}, \text{version}, \text{validity interval}).$$

The adversarial proposal changes at least one bound field:

$$A' \neq A.$$

Therefore the previously earned authority does not transfer. The substituted target must re-enter the lifecycle at the appropriate earlier stage. A prior permit, certificate, or approval cannot be treated as standing trust for a materially different action.

If verification of the new binding returns `FAIL` or `UNDEFINED`, the result is non-authorizing and the action does not proceed.

This example illustrates the operative value of keeping permission, authority, and action distinct: the system can accept that a model generated a candidate while still refusing to let that candidate inherit authority from a different context.

## 12 What Collapse Is Avoided

The authority lifecycle avoids several systemic collapses.

- **Model-output collapse:** plausible output does not become action merely because the model produced it.
- **Self-authorization collapse:** the component seeking to act does not define the rules by which its own authority is judged.
- **Standing-trust collapse:** prior success does not become permanent authority.
- **Missing-data collapse:** absent bindings or evidence do not become implicit defaults.
- **Verification-policy collapse:** validity facts remain distinct from policy action selection.
- **Audit collapse:** execution state is not accepted as sufficient explanation for why an action was allowed.

These distinctions are especially important in AI systems because fluent generation can make missing authority boundaries difficult to see. A system may appear coherent even when it has silently crossed from proposing into governing.

## 13 Provenance and Scope

This paper synthesizes public-safe concepts developed across Logician Regime Theory, DIF, IntegrityGuard, SyNTHe, SWARM, CSP, and Governance Gates. The synthesis is architectural rather than implementation-disclosing: proprietary implementation details, unpublished mathematics, and system-specific enforcement mechanisms remain outside scope.

The purpose of naming that lineage is not to make those project terms prerequisites for the argument. It is to make clear that the authority lifecycle presented here is a synthesis of previously developed burdens, boundaries, and enforcement structures, recast as one public model for AI-mediated action.

## 14 Boundaries and Non-Claims

This paper makes a limited architectural claim.

It does not establish that a particular model is safe, aligned, truthful, or correct.

It does not prove the correctness of evidence sources.

It does not prescribe one cryptographic system, policy language, model family, storage layer, or deployment architecture.

It does not claim that authorization controls solve all AI safety problems.

It does not claim that verification of declared conditions is equivalent to verification of general intelligence or intent.

It does not establish a complete theory of revocation, delegation, or recovery latency.

The result is narrower: where AI output can become consequential action, authority should be explicit, typed, bounded, independently governed, auditable, revalidated, and revocable.

## 15 Result

The paper synthesizes an AI-specific authority lifecycle from previously separate ideas about inquiry, constitutional burden, verification, policy, and assurance.

The result can be stated as follows:

An AI system should not be authorized to act because a model is capable, persuasive, authenticated, or previously trusted. A governed action should occur only when the present authority, scope, evidence, verification state, and policy conditions have been established for that action.

This is why authority must be earned.

The phrase does not mean that a model accumulates moral standing or generalized trust. It means that a particular governed action acquires a bounded basis for execution only after the required burdens have been carried.

## 16 Next Pressure

Several questions remain open.

How should authority propagate through delegated agents without becoming transitive standing trust?

How should revalidation cadence be chosen when model, environment, policy, and external state change at different rates?

How should revocation propagate across long-running plans and partially completed actions?

What minimal audit record is sufficient for independent reconstruction across implementations?

How should systems measure false authorization, false rejection, revocation latency, audit completeness, and recovery quality under adversarial conditions?

Those questions are not answered here.

They are the next pressure created by the present recovery.

## References

- [1] J. H. Saltzer and M. D. Schroeder, “The Protection of Information in Computer Systems,” *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

- [2] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, *Zero Trust Architecture*, NIST Special Publication 800-207, National Institute of Standards and Technology, 2020.
- [3] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, 2023.
- [4] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “ReAct: Synergizing Reasoning and Acting in Language Models,” *International Conference on Learning Representations*, 2023.
- [5] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, “Toolformer: Language Models Can Teach Themselves to Use Tools,” *Advances in Neural Information Processing Systems*, vol. 36, 2023.
- [6] Z. Porter, I. Habli, J. McDermid, et al., “A principles-based ethics assurance argument pattern for AI and autonomous systems,” *AI and Ethics*, vol. 4, pp. 593–616, 2024.
- [7] R. Adler and M. Klaes, “Assurance Cases as Foundation Stone for Auditing AI-enabled and Autonomous Systems: Workshop Results and Political Recommendations for Action from the ExamAI Project,” arXiv:2208.08198, 2022.
- [8] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection,” arXiv:2302.12173, 2023.
- [9] 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,” *Advances in Neural Information Processing Systems*, vol. 37, 2024.