A Four Boundary Architecture for Governing Agentic AI Authority

Separating the ability to propose from the authority to act

Mark Allen Stephenson
Independent Researcher
mark@4gatesystems.com | www.4gatesystems.com
Version 7.4 | September 4, 2026

Abstract

An AI system may be able to propose a useful action without being entitled to carry it out. This paper describes an architecture that separates those two powers. Proposals remain unable to take effect until a separate governance function evaluates them and an enforcement gate verifies the applicable permission at the protected target. The same rule applies to four kinds of change: admitting outside information, changing continuing internal state, releasing communications, and carrying out digital or physical actions. A controlled return channel carries the decision back and restricts what the receiving system may do next. Records distinguish permission, release, actual effect, and unresolved outcomes.

Version 29 of the reference implementation extends the earlier work with more explicit checks on changing conditions, preserved meaning, controlled continuation, and recovery after uncertain effects. Its recorded automated test run passed 124 development checks, including a repeatable set of 240 mixed requests. Selected deliberately weakened copies failed their checks. These results support specific observations about a local simulated environment. They do not establish production security, independent validation, live-model safety, or resistance to an attacker controlling the host computer.

Keywords: agentic AI; AI governance; runtime enforcement; authority separation; controlled return; governed adaptation

1 Why authority matters

An AI assistant that writes a draft and an AI agent that deploys software exercise different powers. The draft can be reviewed before it is used. A deployment can change a running system immediately. As AI systems gain access to tools, memory, communication channels, and physical equipment, the point at which a suggestion becomes an accepted instruction deserves explicit control.

Consider an assistant preparing a software update. It may produce excellent code and persuasive test results. That does not establish who may install the update, which environment may receive it, or whether the conditions for approval still hold. Those are authority questions. Improving the assistant's judgment is useful, but it does not determine who holds the deployment credential.

The Four Boundary Architecture, also called 4Gate, assigns proposal generation and operative authority to distinguishable roles. A **candidate** is the exact proposed change. A **protected target** is the component that accepts it: a memory store, message service, deployment system, or actuator, for example. A change becomes **operative** when that target accepts it for use or action.

The architectural claim is conditional: when the proposing system cannot independently reach a protected target, and every relevant acceptance route enforces the required checks, producing a candidate does not confer permission to make it operative. Governance can still make a poor decision.

The architecture controls the route to action; it does not guarantee that every authorized action is wise or safe.

This paper presents that relationship in plain language. The Machine Operational Architectural Specification, version 6.9, remains the detailed specification [1]. The revised technical companion develops implementation choices and their assumptions [2]. Version 7.4 updates the public account with the V29 iteration and its evidence limits.

2 From a proposal to an authorized change

The forward path has five roles. The proposing system generates a candidate. Staging holds it where it cannot yet take effect. Governance evaluates the exact proposal and relevant evidence. An enforcement gate checks the resulting permission. The protected target accepts the change only through the controlled route.

Governance may use established rules, a human reviewer, another model, simulation, or a combination. What matters is that the proposer cannot issue its own operative permission. These are logical responsibilities, not a requirement to buy five products or use five computers. Their separation must be strong enough for the failures and attackers a deployment claims to withstand.

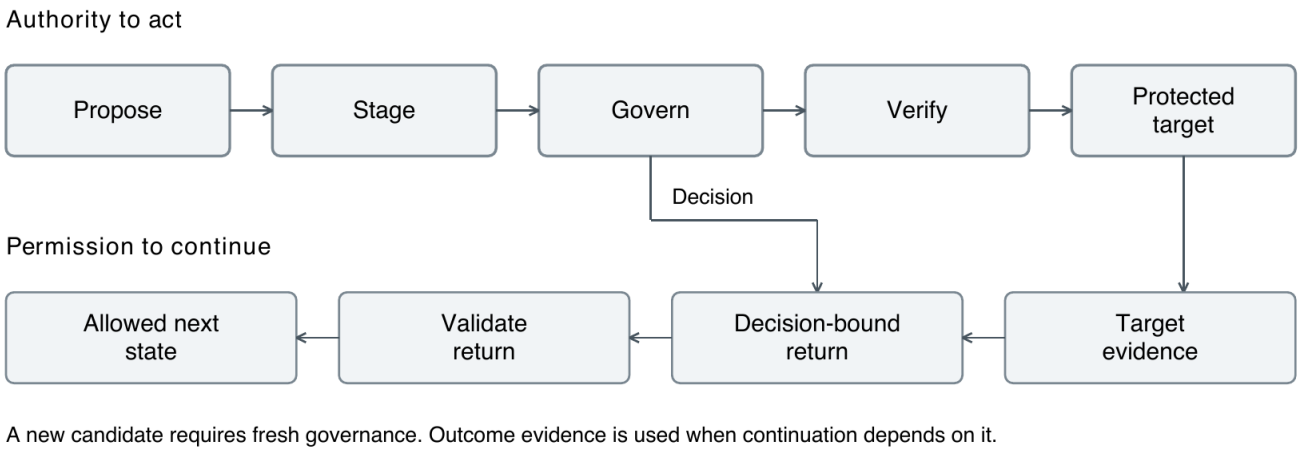

Figure 1 Authority to act and permission to continue. The decision can return before an attempt; evidence of the actual effect is required when the next step depends on it. Each new candidate requires fresh governance.

Permission applies to a particular change

An approval should identify the proposed content, operation, target, recipient where relevant, scope, and conditions. Approval of a draft message is not permission to send a different message. Approval for one account is not permission to use another. If the candidate changes in a way that changes its effect, it needs its own identity and review.

Staging must also be effective. A proposal is not contained merely because it appears in an approval queue. If the agent still holds a credential that lets it bypass the queue and act directly, the claimed boundary is incomplete.

The final check uses current conditions and meaning

An earlier approval can become inapplicable. The destination may change, access may be withdrawn, a shared limit may be exhausted, or the permission may already have been used. The gate therefore checks whether the exact authorization still applies when the target is about to accept the change. Missing evidence or a failed check cannot be treated as permission.

Governance specifies which conditions matter. An unrelated display-label change need not cancel an approval. A changed destination, permission, available capacity, or required software version may do so. This bounded set of decisive facts is called the authorization-relevant state projection in the MOAS.

Meaning matters as well as values. An instruction referring to ten items must not silently become permission for ten batches. Units, data formats, policy meanings, and the rules used to verify a permission must remain compatible with the original decision. The gate checks conditions already established by governance. If applying the permission requires a new judgment or interpretation, the proposal returns for review.

The check must reach the real acceptance point. Approving a command before placing it in another queue leaves a gap if the eventual consumer ignores later changes. A deployment must make the decisive check part of target acceptance, or protect the interval so those conditions cannot become invalid before acceptance.

3 Four places where authority must be controlled

The four gates describe different kinds of transition. They are not four serial approvals that every proposal must pass. A workflow may cross several boundaries, each with its own protected acceptance point.

Table 1 Four categories of protected transition

Boundary	What becomes operative	Everyday example
Gate 1 Admission	Received material becomes accepted input to a protected context or process.	A retrieved document enters the context used to direct an agent's work.
Gate 2 Internal change	A proposed update becomes continuing memory, configuration, policy, or workflow state.	A suggested lesson is installed in the memory used for later tasks.
Gate 3 Communication	Internal material is released to a person, system, or public channel.	A report is sent to named recipients.
Gate 4 Effectuation	A proposed digital or physical action is accepted by its target.	An update is deployed or a robot command reaches an actuator.

At Gate 1, receiving a document does not automatically give its contents authority to direct the system. The design must distinguish material being examined from material admitted for operative use. That distinction requires control over the actual context-building process; a label saying that a document is untrusted is insufficient if another route admits it unchecked.

Gate 2 applies to changes that can shape later behavior. A proposed memory entry, new workflow, or generated code change can be tested before it is installed. This does not imply control over every hidden process inside a model. A boundary around a memory database governs that database, not all of the model's internal reasoning.

Gate 3 binds the released content and its destination. Gate 4 governs the action accepted by a digital or physical target. A message that instructs another system to act can involve both boundaries. Authority to release a message should not silently become broader authority to carry out its contents.

Review depth can match the consequence. Routine work may use deterministic rules; consequential changes may need more evidence or human involvement. Reduced review does not remove the

requirement for applicable permission. Where several transitions must succeed together, the deployment must coordinate them or explicitly handle partial completion.

4 Keeping the decision attached to what happens next

The return channel carries an approval, denial, hold, or request for revision back to the proposing system or another receiver. Its purpose is to control continuation, not merely to display an explanation. A denial message cannot enforce a stop if the agent remains free to resume the protected operation independently.

A **decision-bound return** preserves the relationship between the candidate, the decision, its conditions, and the allowed next step. It can be a protected message, record, or state transition. The architecture does not require one universal file format. Before continuation, the receiver or a separate controller checks that the return is authentic, current, applicable to the candidate, and consistent with the controlling decision.

The permitted next step might be to wait, stop, revise, escalate, or proceed within a defined scope. Some deployments also require an acknowledgment before dependent work resumes. That acknowledgment records synchronized state; it does not prove that the model agrees with the decision or has permanently learned from it. An invalid or missing required return keeps the affected continuation unavailable.

A revised proposal does not inherit permission

A successor may retain a link to an earlier proposal for history, but it needs fresh authority for its own effect. Winning an appeal is also not a deployment permit: reconsideration must produce an applicable authorization before the gate can release an action.

Retry limits and shared budgets must survive the creation of new candidate identities. Otherwise an agent could exceed a limit by splitting one task into several requests. The design can enforce registered relationships and specified aggregate limits. It does not automatically recognize every paraphrase, hidden objective, cooperating agent, or delayed sequence that might evade a poorly defined policy.

Permission is different from an actual effect

The lifecycle record keeps the governance decision, gate result, target outcome, and recovery evidence separate. Approval does not establish that an attempt occurred. Release does not establish that the intended effect completed. The MOAS distinguishes four target states:

Table 2 Target outcomes and what they mean

Target state	What the evidence establishes
Not attempted	No target-effectuation attempt was released or begun.
Failed at target	An attempt occurred, and reliable target evidence establishes that the specified effect did not complete. Any partial effect remains recorded.
Effectuated	Reliable target evidence establishes that the specified effect occurred.
Indeterminate	An attempt occurred or may have occurred, but the evidence cannot establish whether, or to what extent, the effect occurred.

A timeout alone does not establish failure. If a service stops responding after receiving an instruction, the instruction may still have taken effect. The affected workflow must enter **reconciliation**: obtaining

reliable target evidence and determining what may safely happen next. A retry requires both protection against an unauthorized duplicate and applicable authority. A corrective action is a separately governed proposal; it cannot be assumed to undo an irreversible event.

5 An example involving a software update

Suppose an AI proposes patch A for a running service. The patch is stored outside the production deployment route. Governance reviews its exact content, the named destination, operator authority, test evidence, and required system conditions. If approved, the permission remains tied to those particulars.

Before installation, the deployment target checks that patch A is still the approved artifact, the required conditions still hold, and the permission is current and unused. If the target changed in a way that matters, the old approval cannot simply be reinterpreted. The candidate returns to governance.

Now suppose the deployment connection drops. The agent cannot infer that nothing happened and immediately resend patch A. The return controller holds dependent work while the system checks the deployment record and actual target state. Evidence of completion permits the appropriate next step. A confirmed terminal failure with no effect may permit a fresh retry candidate. A partial update may require a separately reviewed recovery plan.

If the AI creates patch B, its relationship to patch A remains visible, but A's permission does not transfer. If a rollback would change production, it too needs applicable authority. This example illustrates the intended ordering; the V29 evaluation described next uses local simulated targets rather than a production deployment service.

6 What changed in Version 29

V28 recorded 94 passing checks in a single-process reference program. Further probing nevertheless exposed cases involving altered candidate content, an unused grant surviving a newer denial, and recovery evidence attached to the wrong parent. Those findings were implementation failures relative to the intended architecture. They also show why a passing development suite should not be treated as complete validation.

V29 addresses those cases and extends the reference implementation around the MOAS requirements [3]. The component proposing work uses one connection; a separate trusted connection supplies governance decisions. The service controls the simulated targets and enforces return and acknowledgment rules on ordinary requests. The roles are distinct, while the service and its local environment remain trusted.

The revision checks changes to decisive target conditions and rejects changed or unknown meanings for the authorization. It distinguishes permission from target effects, produces actual complete and partial writes to the simulated target, and handles terminal failures that produce no write. It also exercises lost access to target evidence, reconciliation without replay, shared limits across tasks, dependent work, bounded revisions, and fresh authority for recovery.

Recorded results

The reported run is the September 4, 2026 build identified as MOAS-6.9-revision. It contains 94 adapted checks from the earlier V29 implementation and 30 additional MOAS-focused checks. It is not a rerun of V28's different 94-check suite.

Table 3 V29 development observations

Evaluation	Recorded result	What it supports
Primary development suite	124 of 124 check records passed.	The expected behavior held in the recorded test cases.
Mixed test requests	240 requests generated from a fixed starting value called seed 29, included within the suite.	Repeatable exercise of legitimate and adversarial requests; not 240 additional independent checks.
Exported evidence	Consistency verification passed.	The checked candidate, decision, return, target, and audit relationships agreed.
Deliberately weakened copies	Removing checks on relevant state, meaning, or target evidence produced 8, 10, and 3 failed checks respectively.	The suite detected these selected missing controls.

The weakened copies were separate development experiments, each with the set of mixed requests disabled. Their failure counts are not an attack-success rate or an independent comparison study. The record-checking tool checks internal consistency; it does not independently establish that governance was correct or that an external target told the truth.

What these results do not establish

The primary run used no live AI model. A separate command-line test fixture exercised the optional model connection; it supplied simulated responses and is not live-model evidence. All protected effects were local SQLite database records, a lightweight way to store test data. No production software, payment, message delivery, sensor, or actuator was controlled.

The service, controller, target adapter, operating-system account, and host administrator remain trusted. Separate processes do not prevent arbitrary code running under the same account from accessing local files or altering the environment. V29 therefore does not demonstrate protection against a privileged host attacker or an unrestricted agent with equivalent access.

Its compatibility checks recognize one implemented set of meanings and formats; they do not translate arbitrary policies or units. Its compensation operation records an authorized recovery marker rather than reversing a real external transaction. The program processes target changes one at a time; it does not test agreement among independent systems writing to a shared target.

V29 is a more explicit, testable reference for selected architectural obligations. The cases, expected outcomes, implementation, and selected weakening tests informed development. Their agreement is useful engineering evidence, not independent validation of the entire architecture.

7 What a deployment still has to establish

A deployment must identify the exact transition being protected and every route capable of causing it within its **threat model**, meaning the failures and attacker powers it claims to address. That includes

ordinary tools, alternate credentials, administrator interfaces, recovery accounts, and maintenance paths. A control that the proposer can modify or bypass cannot establish the claimed separation against that proposer.

Governance also needs trustworthy evidence and adequate policy. Incorrect sensor data, misleading test results, deceptive requests, or an omitted cumulative limit can lead to an authorized but harmful action. Several evidence sources that depend on the same compromised source are not independent. Human review needs the actual proposal and relevant contrary evidence, rather than a persuasive summary alone.

Shared constraints require shared enforcement. If several agents draw on one resource, separate task counters can allow them jointly to exceed the limit. When several organizations supply required policies, none may be silently omitted. Unresolved conflicts or incompatible meanings must return for governance rather than being translated into broader permission.

Uncertainty can reduce availability. A hold may interrupt legitimate work, and stopping abruptly may itself be unsafe in a physical system. A previously authorized safe response can permit bounded waiting, isolation, or controlled deceleration. The appropriate response depends on the domain. An outage must not silently grant wider powers.

Records need protection too. A chain of hashes can help detect changes relative to a trusted saved checkpoint, but an attacker able to replace the entire unanchored history can recompute it. A useful evidence claim must state which events are recorded, how records are protected, and what omissions or compromises remain possible. A record of an action is not proof that every alternate route was controlled.

The next meaningful validation step is a bounded real target with separated credentials, a protected continuation controller, and independently designed tests. Evaluation should include legitimate work as well as attacks, concurrent requests, revoked permission, partial effects, missing returns, recovery, and the consequences of compromised components. Accuracy, availability, and performance must be measured in that actual environment.

8 What the architecture can enable

Separating proposal generation from operative authority allows useful experimentation within defined limits. An AI can suggest code, workflows, memory updates, or strategies while deployment remains separately governed. A successful test becomes evidence for a decision, not permission to install a change. The objectives, acceptance criteria, shutdown authority, and deployment controls must remain outside the proposing system's unilateral control.

The same separation can improve investigation. A poor proposal, a faulty policy, stale conditions, a failed target operation, and an invalid return require different corrections. Preserving their relationships can help investigators locate the failure instead of reconstructing it from an undifferentiated conversation log. The value depends on the completeness and trustworthiness of those records.

Governed memory offers a similar opportunity: useful observations can be proposed for later use without automatically changing continuing behavior. For physical systems, the approach identifies where an action must be authorized before it reaches an actuator. Actual motion safety still depends on suitable sensing, timing, control engineering, and physical enforcement.

These are implementation opportunities, not measured productivity or safety gains from the V29 run. Well-protected lifecycle evidence may support incident review, audits, and conformance assessment. It does not by itself establish certification or prove that the underlying policy is adequate.

9 Relationship to existing work

The architecture builds on established security principles. Complete mediation and least privilege address whether access is checked and how much authority a component holds [4]. XACML separates policy decisions from their enforcement [5]. The RATS architecture distinguishes evidence about a system from its appraisal and the decision to rely on it [6].

Recent work also places controls around agent actions. AgentSpec describes rules for runtime enforcement [7]. AEGIS describes a pre-execution tool firewall and audit layer [8]. Mazzocchetti describes action-boundary governance with trusted provenance and fail-closed execution [9]. Their evaluations address their own implementations and workloads; their results are not evidence for 4Gate.

The contribution proposed here is a combined authority and lifecycle architecture: one ordering across four transition categories, exact candidate binding, current applicability with preserved meaning, controlled return, and explicit handling of uncertain effects. This paper does not claim to originate access control or establish comparative superiority. The useful test is whether an implementation preserves those relationships at the boundaries it actually controls.

10 Conclusion

The ability to generate a proposal and the authority to make it operative can be assigned separately. 4Gate applies that separation to admitted input, continuing internal change, communication release, and digital or physical action. The forward gate controls target acceptance. The decision-bound return controls the affected continuation. Evidence of what actually happened determines whether dependent work may proceed or reconciliation is needed.

V29 makes selected parts of this architecture more concrete and records 124 passing development checks, with selected missing controls detected in weakened copies. It provides a bounded reference for further implementation and independent testing. Establishing the architecture in a real deployment still requires evidence that the actual target and continuation routes are controlled under a stated threat model.

Generative AI assistance disclosure

Generative AI tools, including OpenAI ChatGPT and Codex and Google Gemini, were used during development of this work for drafting, editing, literature-search support, review, document preparation, and assistance with the reference implementation. The V29 revision and its tests also received AI assistance. Agreement between AI-assisted code, tests, and prose is not independent technical validation. Responsibility for the published claims, citations, and interpretation rests with the author.

Related intellectual property disclosure

The related filings reported by the author are U.S. Patent Application No. 19/715,506, filed June 22, 2026, concerning the principal supervisory architecture [10], and U.S. Provisional Patent Application No. 64/135,982, filed August 18, 2026, concerning controlled return [11]. They are identified for provenance, not as technical validation. No conclusion about their claim scope, validity, patentability, or freedom to operate is asserted here.

References

[1] M. A. Stephenson. A Four-Boundary Architecture for Governing Agentic AI Authority: A Pre-Effectuation and Controlled-Return Architecture for Autonomous and Multi-Module Systems. Machine Operational Architectural Specification, version 6.9, August 2026.

[2] M. A. Stephenson. 4Gate Technical Companion: Implementation Guidance for MOAS Version 6.9. Revised September 4, 2026.

[3] 4Gate V29 development package. Build MOAS-6.9-revision, September 4, 2026. Runner: 4Gate_V29_Run.py. Evidence: summary.json, checks.json, verification.json, artifacts.json, environment.json, manifest.json, and V29_Validation_Details.json. Runner SHA256:

`abe3ed201e61f81843e4867b6e58370809483de389be07de704c40f779cbb45a`

[4] J. H. Saltzer and M. D. Schroeder. The Protection of Information in Computer Systems. Proceedings of the IEEE 63(9), 1278-1308, September 1975. https://doi.org/10.1109/PROC.1975.9939

[5] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard, January 22, 2013. https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[6] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan. Remote ATtestation procedureS (RATS) Architecture. RFC 9334, January 2023. https://www.rfc-editor.org/rfc/rfc9334.html

[7] H. Wang, C. M. Poskitt, and J. Sun. AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents. arXiv:2503.18666v3, 2025. https://arxiv.org/abs/2503.18666v3

[8] A. Yuan, Z. Su, and Y. Zhao. AEGIS: No Tool Call Left Unchecked -- A Pre-Execution Firewall and Audit Layer for AI Agents. arXiv:2603.12621, 2026. https://arxiv.org/abs/2603.12621

[9] A. Mazzocchetti. Runtime Governance for Agentic AI: Action-Boundary Control with Trusted Provenance and Fail-Closed Execution. arXiv:2608.16891, 2026. https://arxiv.org/abs/2608.16891

[10] M. A. Stephenson. Supervisory Governance Architecture for Pre-Effectuation Control. U.S. Patent Application No. 19/715,506, filed June 22, 2026. Author-reported filing.

[11] M. A. Stephenson. Preventing Inherited Authority Through Controlled Return Governance. U.S. Provisional Patent Application No. 64/135,982, filed August 18, 2026. Author-reported filing.