HARTSTONE INSTITUTE

WHERE IDEAS BECOME INFRASTRUCTURE

CATEGORY PAPER

Runtime Authority Infrastructure

Governing and Proving AI-Initiated Execution

Emily Hartstone

FOUNDER, HARTSTONE INSTITUTE
FOUNDER, RUNTIME AUTHORITY CONTROL

A HARTSTONE INSTITUTE CATEGORY PAPER

WHERE IDEAS BECOME INFRASTRUCTURE
JUNE 2026

Executive Summary

Artificial intelligence is moving from output to execution.

The first wave of enterprise AI focused on what systems could generate: answers, summaries, recommendations, code, creative assets, analysis, and decisions. The next wave is different. AI systems are beginning to initiate actions inside real environments. They can send communications, update records, trigger workflows, call APIs, move data, create assets, escalate tickets, approve requests, and interact with systems that create operational, legal, financial, reputational, and regulatory consequences.

This shift exposes a governance gap.

Identity systems determine who can access a system. Policy systems define what rules exist. Monitoring systems record what happened. But AI-initiated execution creates a new question:

Should this action be allowed to occur right now, under this identity, in this context, against this system, under these rules?

That question cannot be answered by access alone. It cannot be answered by policy documents sitting outside the execution path. It cannot be solved by dashboards that show risk after the fact. And it cannot be reduced to model governance, because the issue is not only what the AI says or predicts. The issue is what the AI is authorized to do.

Hartstone Institute defines the emerging category as Runtime Authority Infrastructure.

Runtime Authority Infrastructure is the control layer that governs machine-initiated and AI-initiated execution before enterprise state changes occur. It evaluates execution authority in real time, produces deterministic decisions, and creates decision-grade evidence showing what was authorized, constrained, or blocked, under which rules, and when.

This category is not only about enforcement. It is about accountability.

Runtime authority is incomplete if it only blocks or allows an action. A mature system must also prove that the decision was valid. It must preserve evidence. It must support verification over time. It must answer the questions executives, regulators, auditors, insurers, security teams, and boards will increasingly ask:

> Who authorized the execution?
>
> Which rules governed the decision?

> Was the action allowed, constrained, or blocked?
>
> Where is the evidence?
>
> Can the organization prove the control remained true over time?

The future of AI governance will not be defined by who has the most policies. It will be defined by who can govern execution before it happens and prove that governance after it occurs.

That is the purpose of Runtime Authority Infrastructure.

1 The Shift: From AI Output to AI Execution

For years, enterprise AI conversations centered on outputs.

> Can the model answer a question?
>
> Can it summarize a document?
>
> Can it classify risk?
>
> Can it draft copy?
>
> Can it write code?
>
> Can it help employees move faster?

Those questions still matter. But they are no longer the edge of the problem.

The new enterprise risk begins when AI systems move from generating information to initiating action. Once an AI system can trigger a workflow, call an API, modify a record, send a message, publish an asset, update a customer profile, issue a refund, approve a transaction, or interact with a live system, the governance problem changes.

The enterprise is no longer managing content alone.

It is managing execution authority.

That distinction matters because execution creates consequences. A generated answer may be wrong. An executed action may be irreversible, externally visible, financially material, contractually binding, or legally discoverable.

When AI systems gain operational reach, the organization needs a way to determine whether a specific action should occur before it occurs. The enterprise must move from passive review to runtime authority.

This is where current control models begin to strain.

2 The Governance Gap

Most enterprise control stacks were designed around human-initiated execution.

A human employee logs in. An identity provider authenticates that person. Role-based permissions determine what systems they can access. Policies define what they are supposed to do. Logs capture activity. Security tools monitor events. Audit teams review evidence after the fact.

That model assumes the initiating actor is human, intentional, accountable, and operating inside a known chain of responsibility.

AI changes that assumption.

AI agents, automations, copilots, orchestration systems, and embedded intelligence can initiate actions across multiple systems. Those actions may be triggered by prompts, workflows, model outputs, third-party integrations, scheduled routines, or chained agent behavior. They may operate under service accounts, delegated credentials, API keys, or enterprise identities originally designed for human or application access.

Identity answers one question: Who or what has access?

Policy answers another: What rules exist?

Monitoring answers another: What happened?

But AI execution requires a different question:

Was this specific action authorized at the moment of execution?

That is the missing layer.

Without runtime authority, organizations may know that an AI system had access. They may know that a policy existed somewhere. They may even know that an action occurred. But they may not be able to prove that the action was validly authorized before it changed enterprise state.

That is not a logging problem.

That is an authority problem.

3 Why Existing Categories Are Not Enough

The market already has adjacent categories that solve important parts of the broader governance landscape.

Identity and access management systems govern authentication, authorization, and access rights. They answer who can enter which systems and under what permissions.

Policy engines help define and evaluate rules. They can express logic, conditions, and enforcement decisions in structured ways.

API security platforms monitor, protect, and analyze API traffic.

AI governance platforms focus on model risk, compliance, evaluation, documentation, safety, monitoring, bias, explainability, and lifecycle governance.

SIEM and observability platforms aggregate logs, detect anomalies, and support investigation.

Each of these categories is valuable. None of them fully owns runtime authority for AI-initiated execution.

The gap is not that enterprises lack tools. The gap is that the tools were built around different control questions.

Runtime Authority Infrastructure asks:

> Should this AI-initiated action execute right now?
>
> Does the actor have authority for this action, not merely access to the system?
>
> Does the requested execution violate policy, context, role, environment, resource, or operational constraints?
>
> Should the action be allowed, constrained, or blocked?
>
> What evidence proves the decision was valid?

This is a distinct control plane.

It sits between AI systems and enterprise execution. It does not replace identity. It does not replace policy. It does not replace audit. It connects them at the moment authority matters most: immediately before execution.

4 Defining Runtime Authority Infrastructure

Runtime Authority Infrastructure is the enterprise layer that governs AI-initiated and machine-initiated actions before they create state change.

It evaluates execution requests in real time against identity, role, action, resource, policy, environment, and operational context. It produces deterministic decisions. It creates audit-ready evidence for every governed decision.

A runtime authority layer should be able to answer:

> Who or what is initiating the action?
>
> What role or authority does the actor claim?
>
> What action is being requested?
>
> What resource or system will be affected?
>
> Which policy bundle or control rules apply?
>
> What operational context matters?
>
> Should the action be allowed, constrained, or blocked?
>
> What proof exists that the decision was made correctly?

This is different from traditional access control.

Access control may determine whether a service account can call an API. Runtime authority determines whether the specific AI-initiated call should be permitted under the conditions present at that moment.

This is also different from after-the-fact monitoring.

Monitoring may detect that a risky action occurred. Runtime authority determines whether the action should occur at all.

A simple way to express the category is:

Identity governs access. Policy defines intent. Runtime authority governs execution. Audit preserves proof. Continuous verification confirms the control remains true.

That final sentence is important because enforcement alone is not enough.

A control layer that cannot prove its decisions will not satisfy the accountability demands of enterprise AI. Runtime authority must produce evidence, and that evidence must be verifiable over time.

5 The Emerging Enterprise Control Stack

As AI execution becomes more common, the enterprise control stack needs to evolve.

A modern AI execution control stack can be understood in four layers:

Identity Layer

Determines who or what is authenticated and what baseline permissions exist.

Policy Layer

Defines the rules, constraints, governance requirements, and organizational intent.

Runtime Authority Layer

Evaluates specific execution requests in real time before enterprise state changes occur.

Execution Layer

Includes APIs, workflows, data systems, communications systems, financial systems, enterprise applications, and operational infrastructure where actions happen.

This stack creates a cleaner separation of responsibility.

Identity should not be forced to answer whether an AI-generated action is contextually appropriate. Policy should not remain disconnected from execution. Audit should not be limited to after-the-fact recordkeeping. Execution systems should not be expected to interpret governance intent on their own.

Runtime authority becomes the missing connective layer.

It translates policy into execution decisions at the moment of action. It governs the boundary between intention and consequence.

6 Enforcement Is Only Half the Problem

The obvious function of runtime authority is enforcement.

A system requests an action. The runtime authority layer evaluates the request. The decision is returned. The action is allowed, constrained, or blocked.

That is necessary, but it is not sufficient.

The larger enterprise problem is accountability.

If an AI system initiates an action and something goes wrong, the organization must be able to reconstruct the authority chain. It must show which rules were applied, what decision was made, when the decision occurred, and why the action was allowed, constrained, or blocked.

In traditional systems, logs often serve as evidence that something happened. But ordinary logs are not the same as decision-grade authority artifacts.

A log may say that an API call occurred.

A runtime authority artifact should show that the API call was evaluated before execution, under a specific policy bundle, against a specific actor, action, resource, and context, resulting in a deterministic decision.

That is a higher standard.

Audit without authority is a receipt for what already happened. Authority without audit is an unverifiable black box. Runtime Authority Infrastructure requires both.

7 From Logs to Decision-Grade Evidence

The enterprise needs to move beyond passive logs toward decision-grade evidence.

A decision-grade artifact should preserve the material facts of the authorization event. At minimum, it should capture the actor, role, requested action, resource, target system, governing policy, decision outcome, rationale, timestamp, and traceable evidence of the evaluation.

The most important distinction is timing.

A standard audit log often records activity after the action occurs. A runtime authority artifact records the governance decision before the action is permitted to proceed.

This changes the accountability posture.

Instead of saying, "We saw what happened," the organization can say, "We evaluated the requested execution before it happened, applied the governing rules, made a deterministic decision, and preserved the evidence."

That is the standard enterprises will need as AI systems become more operationally capable.

The question will not be whether the enterprise had an AI policy.

The question will be whether the enterprise can prove that the policy governed execution.

8 The Role of Continuous Verification

Once runtime authority decisions generate evidence, the next layer is continuous verification.

This is where audit evolves from static recordkeeping into control assurance.

A governed execution environment should be able to verify that authority decisions remain aligned with policy over time. It should be able to detect drift, confirm decision integrity, validate that required artifacts exist, and prove that the control layer is operating as designed.

This is the role of continuous governance verification.

In the Hartstone Institute control model, this verification layer is represented by CORTHEM: the layer responsible for validating, proving, and continuously confirming that runtime governance remains true over time.

This matters because a single correct decision is not enough.

Enterprises need confidence that the system continues to govern correctly across thousands, millions, or billions of execution events. They need evidence that policies are applied consistently, constraints are honored, exceptions are visible, and authority artifacts remain intact.

> *Runtime authority governs the moment.*
> *Continuous verification governs the pattern.*

Together, they create execution accountability.

9 The Runtime Authority Accountability Loop

Hartstone Institute views Runtime Authority Infrastructure as part of a broader accountability loop.

The loop begins with detection.

Organizations first need to understand where AI execution risk exists. Which systems can initiate actions? Which workflows are exposed? Which service accounts have authority? Which AI tools can reach operational systems? Which gaps exist between access, policy, and execution?

That diagnostic function is the role of TORIXA: surfacing runtime governance gaps quickly and making the invisible execution surface visible.

The loop then moves to enforcement.

RAC governs AI-initiated execution in real time. It evaluates requests, applies policy, and returns deterministic authority decisions.

The loop then moves to evidence.

Each governed decision produces an authority artifact that shows what was evaluated, what decision was made, and why.

The loop then moves to verification.

CORTHEM continuously verifies that the governance layer remains intact, evidence remains trustworthy, and control behavior remains aligned with policy over time.

Finally, the loop is tested in production.

RIMAGINC acts as a real-world execution environment where AI-generated outputs, brand controls, policy enforcement, and runtime governance can be validated against live product demands.

The closed-loop model is:

TORIXA identifies the gap. RAC governs the execution. Audit preserves the evidence. CORTHEM verifies the control. RIMAGINC stress-tests the system in production.

This is not a collection of disconnected products.

It is an accountability architecture.

10 Why Runtime Authority Matters Now

The urgency comes from a timing mismatch.

AI execution capability is advancing faster than enterprise governance infrastructure.

Organizations are already adopting AI systems that can interact with internal tools, external APIs, customer data, operational systems, creative workflows, financial processes, and business-critical infrastructure. At the same time, most enterprise control stacks still assume that execution authority flows through humans, traditional applications, or predefined workflows.

That assumption is breaking.

As AI systems become more agentic, the enterprise needs a control layer that can govern execution without waiting for a human to manually review every action. But that control layer must also be deterministic enough for compliance, explainable enough for leadership, and auditable enough for regulators, insurers, and internal risk teams.

This is the architectural moment for Runtime Authority Infrastructure.

The market does not need another dashboard that reports risk after the damage is done.

It needs a runtime control plane that governs AI-initiated execution before enterprise state changes occur.

11 The Adoption Model: From Visibility to Enforcement

Runtime Authority Infrastructure does not have to begin with full blocking enforcement.

A mature adoption model should allow organizations to move in stages.

Stage One: Observation

The runtime authority layer observes execution requests, evaluates policy, and produces decision evidence without blocking actions. This allows organizations to understand their execution surface, policy gaps, risky workflows, and operational patterns before enforcing controls.

Observation is valuable because many organizations do not yet know where AI execution authority exists. They cannot govern what they cannot see.

Stage Two: Controlled Enforcement

Once high-risk execution paths are understood, enforcement can begin in targeted areas. These may include financial actions, external communications, sensitive data access, production workflow changes, customer-impacting actions, regulated processes, or brand-sensitive outputs.

Controlled enforcement allows organizations to tune policy, validate behavior, and reduce operational risk.

Stage Three: Full Runtime Governance

Eventually, runtime authority becomes a standard control layer across AI-initiated execution. The organization no longer treats AI execution as an ungoverned side channel. It becomes part of the enterprise control stack.

This staged model matters because enterprises need governance without unnecessary disruption.

The goal is not to slow innovation. The goal is to make execution trustworthy enough to scale.

12 The Difference Between Access and Authority

One of the most important distinctions in this category is the difference between access and authority.

Access means a system can do something.

Authority means the system should do that thing under the specific conditions present at the moment of execution.

An AI agent may have access to a customer database. That does not mean it has authority to export a sensitive customer segment.

A workflow automation may have access to send emails. That does not mean it has authority to send a regulated communication without required context.

A service account may have access to a payment system. That does not mean it has authority to approve a transaction outside defined constraints.

A generative system may have access to publish brand content. That does not mean it has authority to release content that violates brand, legal, or intellectual property requirements.

The access question is binary and structural.

The authority question is contextual and operational.

Runtime Authority Infrastructure exists because AI execution makes that distinction unavoidable.

13 The Board-Level Question

The board-level question is not "Do we use AI?"

Most organizations will.

The better question is:

Can we prove that AI-initiated execution is governed before it creates consequences?

That question will matter to boards, CISOs, CIOs, compliance officers, legal teams, insurers, auditors, regulators, and customers.

> It will matter when AI systems interact with sensitive data.
>
> It will matter when AI systems trigger external communications.
>
> It will matter when AI systems modify operational records.
>
> It will matter when AI systems influence financial actions.
>
> It will matter when AI systems create brand, legal, or customer-facing outputs.

> It will matter when an organization needs to explain not only what happened, but why it was authorized.

The organizations that answer this question early will have a governance advantage.

The organizations that wait will discover the gap during an incident.

That is usually the expensive way to learn architecture.

14 Runtime Authority as Enterprise Infrastructure

Runtime authority should not be treated as a feature inside a single application.

It is infrastructure.

The reason is simple: AI execution will not remain confined to one interface, one vendor, one model, one workflow, or one system. It will spread across orchestration layers, internal tools, SaaS platforms, APIs, service accounts, agents, enterprise applications, and custom workflows.

A control layer that only works inside one application cannot govern the full execution surface.

Runtime Authority Infrastructure must be able to operate across boundaries. It must integrate near the points where execution occurs. It may connect through orchestration platforms, API gateways, workflow systems, event buses, service boundaries, or execution proxies. The exact deployment pattern may vary by environment, but the control principle remains the same:

Evaluate before execution. Preserve proof after decision. Verify continuously over time.

That is the infrastructure thesis.

15 What This Category Is Not

Runtime Authority Infrastructure is not a replacement for identity and access management.

Identity remains essential. Runtime authority depends on identity signals, actor context, and role information. But identity alone does not evaluate whether a specific AI-initiated action should execute in a specific context.

Runtime Authority Infrastructure is not a replacement for policy engines.

Policy remains essential. Runtime authority depends on rules. But policy definitions alone do not guarantee that those rules govern live execution at the moment of action.

Runtime Authority Infrastructure is not a replacement for AI governance.

AI governance remains essential. Model evaluation, safety, compliance, documentation, explainability, and risk management all matter. But model governance does not fully answer execution authority.

Runtime Authority Infrastructure is not a dashboard.

Dashboards can show risk. They do not, by themselves, govern whether an action should occur before it changes enterprise state.

Runtime Authority Infrastructure is not merely audit logging.

Audit is essential, but logs alone do not create authority. The category requires pre-execution decisioning and post-decision evidence.

This distinction is critical.

The enterprise does not need another layer of passive visibility pretending to be control. It needs authority, evidence, and verification.

16 The New Standard: Govern and Prove

The future standard for enterprise AI execution will be simple:

Govern and prove.

> Govern the action before it happens.
>
> Prove the decision after it occurs.
>
> Verify the control over time.

This standard reframes AI governance from documentation to execution accountability.

> It is not enough to say the organization has a policy.
>
> It is not enough to say the system had access.
>
> It is not enough to say logs exist.
>
> It is not enough to say humans can review incidents later.

The organization must be able to show that AI-initiated actions were evaluated at runtime, governed by policy, decisioned deterministically, recorded as evidence, and continuously verifiable.

That is the difference between AI adoption and AI execution accountability.

17 Conclusion: The Control Layer AI Execution Requires

AI is becoming operational.

That means enterprise governance must become operational as well.

The next generation of AI risk will not be limited to inaccurate outputs, hallucinated answers, or poorly documented models. It will include unauthorized actions, inappropriate execution, unverified authority, policy drift, missing evidence, and accountability gaps across live systems.

Runtime Authority Infrastructure is the category built for this moment.

It governs AI-initiated execution before enterprise state changes occur. It produces decision-grade evidence. It supports continuous verification. It gives organizations a way to answer the questions that will define the next era of enterprise AI:

> Who initiated the action?
>
> Was it authorized?
>
> Which rules applied?
>
> Was it allowed, constrained, or blocked?
>
> Where is the evidence?
>
> Can the organization prove the control held over time?

The enterprises that answer these questions will be able to scale AI with confidence.

The enterprises that cannot will be left trusting systems they cannot fully govern.

Runtime authority is not a future luxury. It is becoming enterprise infrastructure.

And the organizations that recognize this early will define the standard everyone else is forced to meet later.