HARTSTONE INSTITUTE

WHERE IDEAS BECOME INFRASTRUCTURE

CATEGORY PAPER

Runtime Authority Fabric™

Governing Authority
Across the Agentic Chain

Emily Hartstone

FOUNDER, HARTSTONE INSTITUTE
FOUNDER, RUNTIME AUTHORITY CONTROL

Executive Summary

AI systems are moving from response generation into machine-initiated action.

They do not only answer questions. They call tools, retrieve context, update systems, trigger workflows, delegate tasks, coordinate with other agents, and create downstream consequence.

This shift changes the governance problem. The first question is no longer only whether a human user has access to a system, or whether a model produced an acceptable output. The operational question is whether machine-initiated action is authorized before consequence. That is the category RAC defines.

RAC™, Runtime Authority Control™, governs machine action at runtime. It evaluates machine-initiated action before consequence and returns an allow, block, or constrain decision with preserved decision evidence.

CORTHEM™, Continuous Governance Verification, proves governance over time. It is not logging. It verifies that governed action, governed decisions, and governed authority chains remain evidenced, accountable, and reconstructable.

Runtime Authority Fabric is the next architecture thesis. **RAF™** governs what happens when authority moves across the agentic chain: from one agent to another, from one context to another, from one tool path to another, or from one consequence path into reversal, correction, or coordination.

The next failure mode is not unauthorized access, which identity controls address. It is not unsafe behavior, which guardrails and agent sandboxes increasingly catch. It is authorized machine action crossing agentic boundaries without a governed authority chain.

> **RAC** governs the authority event.
>
> **RAF** governs the authority chain.
>
> **CORTHEM** proves both held.

This paper defines Runtime Authority Fabric as the architecture for preserving authority boundaries across agents, contexts, handoffs, delegation, coordination, and reversal. It does not replace RAC. It does not replace CORTHEM. It extends the terrain those categories make visible.

1. From AI Output to Machine Consequence

For much of the AI era, governance focused on model output: accuracy, bias, safety, explainability, hallucination, and compliance. Those questions still matter, but they are no longer enough. The market is moving toward AI systems that act.

An AI system may draft and send a message, update a customer record, issue a refund, retrieve sensitive context, call an external API, modify an internal workflow, trigger a deployment, instruct another agent, authorize a sub-agent, or coordinate multiple agents toward a shared outcome.

Once AI systems create consequence, governance must operate before consequence. The question becomes: is this machine action authorized before it happens? That is the foundation of RAC. It is not a narrow permission checker. It is not a one-shot rule filter. It is the runtime authority layer between machine intent and the live system the machine would act on.

But agentic systems introduce a second problem. They do not always act through one isolated event. They may act through handoffs, delegated authority, tool chains, memory, context shifts, agent-to-agent instruction, and reversal paths. In those cases the question becomes larger: what happens when authority crosses from one machine actor, context, or consequence path into another? That is the problem Runtime Authority Fabric names.

2. The Three Categories

2.1 RAC™, Runtime Authority Control™

RAC governs machine-initiated action at runtime. Its job is to determine whether a machine action is authorized before consequence, returning decisions such as allow, block, constrain, or escalate.

The RAC question is: *can this machine action happen now?* More fully, is this machine-initiated action authorized under the applicable authority, policy, scope, context, and consequence conditions before it creates impact? RAC remains RAC even when the runtime decision is stateful — when a grant expires, when scope drifts inside one authorized context, when context integrity is questionable, when a consequence tier requires escalation. Those are not separate categories. They are RAC, governing within a single authorized context.

2.2 CORTHEM™, Continuous Governance Verification

CORTHEM proves governance over time. A log records that something happened; CORTHEM verifies that governed action remained evidenced, accountable, and reconstructable. It proves both governed authority events and governed authority chains. The CORTHEM question is: *can we prove governance held over time?*

2.3 RAF™, Runtime Authority Fabric™

Runtime Authority Fabric governs authority crossing agentic boundaries. It begins when authority must move, transfer, delegate, coordinate, or be preserved across more than one machine actor, context, tool path, approval path, or consequence path.

RAF is not "RAC plus time." It is not "CORTHEM plus orchestration." It is not a rebrand of either. RAF is the architecture that carries governed authority across the agentic chain. The RAF question is: *when authority crosses a boundary, does the governed chain remain intact?*

RAF becomes necessary when one agent delegates to another, when an authorized agent attempts to authorize other agents, when a task crosses into another context, when one tool path creates downstream consequence through another, when an agent coordinates multiple agents toward a shared outcome, when a reversal must be coordinated across actors, or when jurisdiction between agents must be established.

3. The Dividing Principle

The categories remain clear if the dividing principle remains clear. A runtime allow, block, constrain, or escalate decision on one consequential machine action belongs to RAC, even when that decision is stateful. A proof obligation over time belongs to CORTHEM. Authority crossing agents, contexts, handoffs, delegation paths, coordination paths, or reversal paths belongs to RAF.

> **RAC** governs the action.
>
> **RAF** governs the chain.
>
> **CORTHEM** proves the action and the chain.

RAC owns the runtime decision. Is this refund authorized? Is this email send authorized? Has this grant expired? Is this action still inside the original scope? Even when the decision depends on time, state, scope, or context, it remains RAC as long as it is evaluating one machine action inside one authorized context.

RAF owns the authority chain. Agent A delegates to Agent B. A workflow crosses from a sales context into a finance context. An operations agent instructs an infrastructure agent. A reversal requires coordination across finance, support, and legal agents. RAF begins when the question is no longer only "is this action allowed?" but "did authority cross a boundary while remaining scoped and governed?"

CORTHEM owns proof. Can we prove RAC allowed or blocked the action? Can we reconstruct the delegated authority chain? Can we prove a reversal was authorized and completed? CORTHEM proves both RAC-governed events and RAF-governed chains.

4. Why Agentic Systems Create a New Authority Problem

Traditional access control assumes a relatively stable relationship between actor, permission, resource, and action. A user has an account; the account has permissions; the user performs an action; the system records an event.

Agentic systems change that pattern. The actor may be a machine acting for a human. The action may be inferred from a goal. The goal may produce multiple steps. The context may come from retrieval, memory, another agent, or a generated summary. The system may delegate. The delegated agent may call tools. The tools may create consequence. The consequence may require reversal. The evidence may be distributed across agents, systems, and time.

The governance gap is not only that machines may act without access controls. It is that machine authority may cross boundaries without a governed chain. A machine may have access and still lack authority. An authorized agent may lack authority to authorize other agents. A delegated agent may receive more scope than the original agent had. A chain of individually permitted actions may create an unauthorized combined consequence. This is why RAC, CORTHEM, and RAF must remain distinct.

5. Authorized to Act Is Not Authorized to Authorize

One of the most important authority problems in agentic systems is the difference between being authorized to act and being authorized to authorize. An agent may be authorized to perform a task. That does not automatically mean the agent is authorized to grant authority to other agents. This distinction is central to Runtime Authority Fabric.

Authorized to act means an agent has authority to perform a specific action or class of actions under defined scope. A support agent authorized to investigate a billing issue may read billing history, review invoices, and prepare a recommendation. RAC governs whether each consequential action is authorized.

Authorized to authorize means an agent has authority to create, delegate, narrow, or assign authority to another agent. When the support agent authorizes a refund agent to issue a refund, it is not merely acting. It is creating a new authority path for another machine actor. When one agent authorizes another, authority is no longer only being used; it is being created, transferred, or reshaped. Governing that creation is a different problem from governing the action itself.

The Agent Authorization Cascade

An agent authorization cascade occurs when an authorized agent creates authority for additional agents, which may then create authority for still more agents or tools. It can happen intentionally, when a planning agent assigns tasks to research, finance, and legal agents. It can happen operationally, when an incident-response agent delegates to log-

analysis, infrastructure, and deployment agents. And it can happen accidentally, when a general assistant tells another agent to "handle it" and the receiving agent interprets that as authority to act externally.

The core issue is not that delegation exists. Delegation is necessary for agentic systems. The issue is whether an authorized machine actor can generate downstream machine authority without losing scope, jurisdiction, proof, or accountability. Runtime Authority Fabric names this problem.

> *An agent authorized to act is not automatically authorized to authorize other agents. Agent-to-agent authorization should be explicit, bounded, and accountable rather than assumed. RAF governs the delegation chain; RAC governs each consequential runtime action within it; CORTHEM proves both over time.*

6. What Stays RAC

Before defining what RAF governs, it is worth being precise about what it does not. A large class of authority questions are already RAC's, and naming them is what keeps the boundary clean. RAC governs the stateful, single-context case: whether a grant has expired, whether scope has drifted inside one authorized context, whether the context justifying an action is trusted and current, what consequence tier an action carries, when to escalate to a human, and whether a reversal action is itself authorized. These are not a roadmap. They are RAC governing one machine action inside one authorized context.

The billing agent draws the line. A user authorizes an agent to investigate a billing issue; the agent finds a possible overcharge and attempts a refund. While one agent acts inside one authorized context — asking whether refund authority was granted, whether the amount is in scope, whether the authority expired — it remains a RAC decision. The moment that agent authorizes a second agent to issue the refund, authority crosses a boundary, and that is where RAF begins. Everything that stays within one authorized context is already RAC's; RAF governs only what crosses.

7. RAF Develops: Delegated and Multi-Agent Authority

Runtime Authority Fabric begins when authority crosses an agentic boundary. Its most important development is delegated authority: governing how authority moves from one actor to another.

The operations agent

A user tells an operations agent to investigate why checkout is slow. The agent reviews metrics and delegates infrastructure analysis to another agent, which identifies database pressure and attempts to resize production

infrastructure. RAC governs the runtime decision to resize. RAF governs the authority chain that led to it: who granted the original authority, what the original scope was, whether the operations agent was permitted to delegate, whether the infrastructure agent was permitted to receive that authority, and whether "investigate" ever included production modification. The issue is not only whether the final action is authorized; it is whether the authority chain that produced it remained governed.

Agent-to-agent authorization

When one machine actor authorizes another, authority is created, transferred, or reshaped rather than merely used. This is where many future failures will occur — not because organizations forgot access control, but because they allowed authorized agents to become unauthorized authorizers.

Multi-agent jurisdiction

Coordination alone is not governance. A group of agents may coordinate effectively while still violating authority boundaries: one agent assuming it can instruct another, override another, or accept authority that was never valid. RAF introduces jurisdiction as an authority concept: which agent may instruct, override, delegate, accept delegated authority, own the final consequence, escalate to a human, or constrain another agent. This is not ordinary orchestration; it is authority governance across machine actors.

Cross-agent coordination

A chain of individually authorized actions can still create an unauthorized combined consequence. One agent retrieves context, another drafts a message, a third sends it, a fourth updates a record, a fifth notifies finance. Each step is acceptable, yet the combined chain exceeds the original authority. RAF governs the joint effect across agents. The question is whether the chain is authorized, not only the step.

Delegation should narrow by default

When authority moves from one agent to another, the receiving agent should hold equal or narrower authority unless explicit escalation or human approval expands it. A delegated agent should not inherit the full authority of the delegating agent by assumption. This may become one of the defining principles of agentic governance.

8. CORTHEM Develops: Proof for Events and Chains

As RAC becomes more stateful and RAF governs cross-agent authority, CORTHEM's proof model expands without becoming orchestration. A RAC-governed event should be reconstructable: what was attempted, by whom, under what authority and policy, with what context and consequence, what RAC decided, and what evidence was preserved.

A RAF-governed chain should also be reconstructable: where authority originated, where it moved, which agents participated, whether delegation was permitted, whether scope narrowed or expanded, whether jurisdiction was crossed, whether human approval was required, and whether the chain remained governed end to end. CORTHEM is not reduced by RAF. It becomes more important as authority chains grow more complex.

9. Reversal and Correction

In ordinary software, rollback often means restoring a prior state or reverting a transaction. In agentic systems, reversal is harder, because machine actions create external consequence. An email is sent, a refund is issued, a partner is notified, a downstream workflow is triggered, a decision is made that cannot be perfectly undone.

A naive rollback mindset says "reverse the transaction." A governed reversal model treats reversal as an authority problem: whether reversal is allowed, who holds the authority to reverse, whether the original action exceeded authority, and whether the correction itself was authorized and provable. If one agent requests permission to reverse, RAC governs that runtime decision. If the reversal requires finance, support, legal, and a notification workflow, RAF governs the reversal chain. CORTHEM proves the original action, the reversal authority, and the final governance record.

10. The Runtime Authority Fabric Model

Runtime Authority Fabric can be understood as the architecture that governs authority across the agentic boundary. A human or system grants authority. RAC governs machine action before consequence, and the decision becomes a governed authority event that CORTHEM proves over time. While the action stays inside one authorized context, RAC continues governing runtime decisions. When authority crosses agents, contexts, tools, handoffs, or reversal paths, RAF governs the authority chain — with RAC still governing each consequential action inside it, and CORTHEM proving both the events and the chain.

11. What Runtime Authority Fabric Is Not

Not a rebrand of RAC. RAC already defines runtime authority over machine action. RAF exists because agentic systems introduce cross-boundary authority chains.

Not a rebrand of CORTHEM. CORTHEM defines continuous governance verification. RAF creates more complex chains for CORTHEM to prove, but it does not replace proof.

Not workflow orchestration. Orchestration coordinates tasks. RAF governs whether authority remains scoped when tasks move across agents, contexts, tools, approvals, and reversal paths.

Not runtime guardrails, agent sandboxing, or behavioral monitoring. Guardrails filter unsafe behavior. Sandboxes constrain what an agent is able to do. Monitoring records whether behavior matched policy. RAF governs what none of these reach: whether the authority to act, and the authority to authorize, moved across the agentic chain while remaining scoped, jurisdictional, and provable. A fully sandboxed, policy-compliant agent can still authorize another agent it had no authority to empower. That is an authority-chain failure, not a behavioral one, and constraint and detection do not see it.

Not a claim that every future capability is shipped. It is the architecture thesis for the next layer of agentic governance. RAF does not make RAC smaller. It becomes necessary because RAC is right: machine action must be governed at runtime, before consequence.

12. Why This Category Must Be Named Now

The market is no longer only beginning to recognize that AI agents need runtime control. The largest infrastructure and compute vendors are now shipping it: real-time guardrails on agent behavior, per-agent sandbox isolation, agent-to-agent monitoring, and a continuous record of whether behavior matched policy. That recognition validates the timing and proves the problem is real. It does not close the category. Guardrails constrain behavior. Sandboxes constrain capability. Monitoring detects deviation. A continuous record observes. None of them governs authority, and none of them governs the authority to authorize as it moves across agents. That is the layer still unnamed, and it is the layer that decides whether agentic systems can be trusted to act.

As agentic systems mature, the governance problem moves from individual agent action into agentic authority chains: can this agent authorize another agent; can that agent delegate again; can authority cross from investigation into execution; can context from one domain justify action in another; can multiple permitted actions combine into an unauthorized outcome; can reversal be authorized across the chain; can the organization prove all of it?

Without precise language, the market will collapse these problems into vague buckets: AI governance, agent security, identity, authorization, runtime guardrails, agent sandboxing, behavioral monitoring, observability, workflow automation, compliance, audit logging, human-in-the-loop approval. Each matters. None fully names agentic authority crossing boundaries. Runtime Authority Fabric names that layer, and gives organizations language for governing machine authority as it moves across agents, contexts, handoffs, delegation, coordination, and reversal.

13. The Frontier

RAC governs stateful runtime authority today; CORTHEM proves governed events today. Those are the settled foundations. The frontier is RAF — the governance of authority in motion: delegation, jurisdiction, coordination,

and reversal across agents and contexts. The foundations exist now; the work ahead is carrying governed authority across the chain.

Conclusion

Agentic AI turns authority into something that can move. It can begin with a human instruction, be interpreted by an agent, be checked by RAC, be shaped by context, be narrowed by scope, expire, require escalation, be passed to another agent, be used to authorize further agents, cross jurisdictions, coordinate multiple actions, create consequence through tools, require reversal, and must be proven afterward.

Traditional access control was not built for that shape of authority. Ordinary logging was not built to prove that shape of governance. Workflow orchestration was not built to preserve authority across that chain.

> **RAC** governs machine action at runtime.
>
> **RAF** governs authority across the agentic chain.
>
> **CORTHEM** proves both held over time.

Runtime Authority Fabric is not the admission that RAC is incomplete. It is the architecture that becomes necessary because RAC is right. Machine action must be governed before consequence, and as machines become agentic, delegated, coordinated, and consequential, that governed authority must be carried across the chain. That is the purpose of Runtime Authority Fabric.