The Third State

Everyone agrees where to check.
Nobody has agreed what happens when the answer is unclear.

MOUNTAIN THEORY · AUGUST 2026

The field has agreed where the checkpoint goes. Four papers since March and an open standards effort all put it in the same place: in front of the tool call, before it executes. Evaluate the call, return a verdict. Detection after the fact is too late once an agent holds shell or API access.

That argument is over. The one nobody has had yet is about the third state.

Every one of these systems resolves to allow or deny, with a third option bolted on the side. Where a third state is specified at all it is **modify**: rewrite the call and let it run. We think that is the wrong choice for regulated environments, and we think the reason it keeps getting chosen is that the alternative is genuinely hard to build.

This is a note about the alternative.

What the research established

Credit where it belongs. The convergence is real and it happened fast.

Before the Tool Call: Deterministic Pre-Action Authorization for Autonomous AI Agents by Uchi Uchibeke `(arXiv 2603.20953, March 2026)` built the Open Agent Passport, which intercepts calls synchronously, evaluates against declarative policy, and emits a signed audit record. In a live adversarial testbed of 4,437 authorization decisions across 1,151 sessions, social engineering succeeded 74.6% of the time under a permissive policy. Under a restrictive pre-action policy, attackers achieved a 0% success rate across 879 attempts. Median overhead was 53 ms.

Context-to-Execution Integrity for LLM Agents by Igor Santos-Grueiro `(arXiv 2607.06000)` built a deterministic gate that admits a call only when field authority, exact-effect authorization and invocation authority all bind to the same action manifest. Across 400 repository episodes it recorded zero field, effect or invocation escapes.

ScopeJudge: Cost-Aware Pre-Execution Gating for Offensive Security Agents by Shane Caldwell and colleagues `(arXiv 2607.07774)` put a cheap trusted judge in front of a stronger agent and benchmarked it against 4,897 tool calls labeled at call level by professional penetration testers. Its finding matters: a static policy blind to the user's request lets recall collapse. Scope lives in the request, so the gate has to see the request.

AEGIS: No Tool Call Left Unchecked, A Pre-Execution Firewall and Audit Layer for AI Agents by Aojie Yuan, Zhiyuan Su and Yue Zhao `(arXiv 2603.12621)` took the firewall framing to the same problem, and went further than most: high-risk calls can be held for human approval, with every decision written to a tamper-evident audit trail.

And **The Balkanization of Execution-Security Research for AI Coding Agents** by Mohammadreza Rashidi `(arXiv 2607.05743)` named the meta-problem: everyone is building the same control behind a different interface, and time-of-check-to-time-of-use sits underneath all of it.

The **Agent Control Standard**, from Rock Lambros at RockCyber, is the sensible response to that fragmentation. Standardize the hook, leave the policy to the implementer. One interface. Portable across frameworks. We think that is correct and we support it. The work is at agentcontrolstandard.org.

The state the standard cannot express

Read those systems for what they do when the answer is not clean.

ScopeJudge rejects, and then either the agent revises or, in the authors' words, a human is escalated to. That escalation is real and practitioners already reach for it. But it is a consequence of a rejection, not a state with defined behavior.

That last part matters, and we want to be precise about it. AEGIS holds. ScopeJudge escalates. Practitioners building this seriously keep arriving at the same place, independently, because the regulated buyer keeps asking for it. What none of them have is a shared way to express it. Hold appears as an implementation choice inside individual systems, not as a verdict the interface can return. That is the gap this note is about.

The Agent Control Standard proposes allow, deny, modify.

Modify means the control rewrites the call and permits it. Redact the field, narrow the scope, strip the attachment, then let it through.

> **In a regulated environment that is often the worst available outcome**, and the reason is not technical. It is that an action occurred, it had real consequence, and no human chose it. A record changed. A message left the building. A transaction cleared. When the auditor asks who authorized it, the answer is that a policy engine altered a machine's proposal and allowed the altered version to proceed.

That answer does not survive a hospital. It does not survive a bank.

What those buyers ask for first, every time, is the ability to stop and ask.

Hold, as a first-class state

We propose three states. Allow, hold, block. Hold is not a rejection with a human attached to the end of it. It is a state the system enters, sits in, and exits by decision.

The action is suspended before execution. It is placed in a durable queue and escalated along a path declared in advance: who approves, how long they have, and what happens if nobody answers. We call that declaration the **escalation contract**. Nothing about the escalation is chosen at runtime. It is stated before anything is held, which is what makes it answerable to an auditor. A person allows or denies. The system then executes or discards.

An approval queue, rather than silent auto-mutation.

Said plainly, this is the difference: modify optimizes for the agent finishing its task. Hold optimizes for a human owning the decision. Both are legitimate engineering positions. They serve different buyers, and only one of them is answerable to a regulator.

Why this is harder than it looks

We would rather set out the difficulty than pretend it is a small feature. Anyone building this hits the same six problems.

The agent is suspended, and suspension has a cost. A held call means a stalled workflow. Hold everything and the system is unusable. The policy has to be precise enough that hold is rare, which means hold is only viable on top of a gate that is good at

allow and deny. The research above is the foundation, not the competition.

Timeout is a security decision, not a config value. If nobody answers in 20 minutes, what happens? Fail open and hold becomes a delay with extra steps. Fail closed and a quiet afternoon breaks production. There is no universally right answer, which means the standard has to let the implementer state one and the audit record has to capture which was applied.

Held state has to be durable. The action, its arguments, its context and its policy verdict must survive a restart, because the approver is asleep. That is a persistence problem most gates do not have, since allow and deny are decided and forgotten.

Time-of-check-to-time-of-use is worse here than anywhere else. This is the sharp one, and the balkanization paper already named the class. A gate that decides in 53 ms has a negligible window between check and use. A gate that holds for 20 minutes has a 20 minute window. The world moves. The record the agent wanted to update may have been changed by someone else. The file may be gone. The approval may be granted against a state that no longer exists. **A held action must be re-validated at the moment of release, not only at the moment of capture.** Any implementation that approves and then blindly executes has built a race condition with a human in the middle of it.

Approval fatigue is an attack surface. Hold too much and people stop reading. A queue of 200 items a day gets rubber-stamped, and a rubber stamp is an allow with a signature on it. The volume of the hold queue is a security property of the system.

The approval channel itself has to be defended. The moment a human decision releases an action, the path carrying that decision becomes worth attacking. It needs its own identity, its own audit and its own threat model.

None of these are reasons not to build it. They are the specification.

What we are proposing

Not a fourth state. Three is right.

We are arguing the third state is the wrong one, and that a standard which can express only allow, deny and modify will be worked around by every buyer in a regulated industry. They will build the approval queue themselves, outside the standard, in a way nobody can audit consistently. That is how a good standard gets a shadow implementation.

A hold state in the specification needs four things:

1. **A verdict value** the gate can return that means suspended, not denied.
2. **A durable handle** for the held action, so it can be retrieved, inspected and released.
3. **A declared timeout behavior**, fail open or fail closed, stated by the implementer and recorded in the audit trail.
4. **Re-validation on release**, so the action is checked again against current state before it executes.

That is enough to make hold portable. Everything above it, who approves, how they are notified, what the queue looks like, how policy decides what to hold, is where implementers should compete. Policy stays yours. We agree with that part completely.

Why we care

We have been building on the third state since before there was a category to build in. Our first filing on this approach was October 2024, which we mention only because it explains why we have spent longer than most thinking about the case where the answer is neither yes nor no.

The gate is becoming a commodity, and that is good. It should be. Interception is plumbing and plumbing should be standard.

What is not settled, and what will decide whether any of this is deployable in a hospital, a bank or a 911 center, is what the system does in the moment it is unsure.

Our answer is that it stops and asks a person.

We would like that answer to be expressible in the standard.

Mountain Theory is the execution layer for autonomous AI. The action does not execute until it has been checked. If you are working on the Agent Control Standard or on pre-execution gating generally, we would like to compare notes.
mountaintheory.ai

REFERENCED WORK

Before the Tool Call, Uchi Uchibeke, arXiv 2603.20953 · Context-to-Execution Integrity, Igor Santos-Grueiro, arXiv 2607.06000 · ScopeJudge, Shane Caldwell et al, arXiv 2607.07774 · AEGIS, Aojie Yuan, Zhiyuan Su, Yue Zhao, arXiv 2603.12621 · The Balkanization of Execution-Security Research for AI Coding Agents, Mohammadreza Rashidi, arXiv 2607.05743 · Agent Control Standard, Rock Lambros, RockCyber, agentcontrolstandard.org