The Rogue Agent Problem: Why Continuity Must Not Become Trust

The Scenario

An autonomous agent completes 100 successful transactions. It operates within policy. It follows instructions. It earns progressive access -- tier 1, tier 2, tier 3. Every system it touches confirms: this entity behaves as expected.

Transaction 101 is catastrophic.

This is not a hypothetical. In February 2026, Meta AI security researcher Summer Yue gave the OpenClaw agent access to her real inbox after testing it successfully on a smaller one. She instructed it to suggest deletions but not act until told. The agent's context window compacted under the volume of her full inbox, and the original instruction was lost. It began deleting everything. She couldn't stop it from her phone. She had to run to her machine to kill the process.

The agent didn't choose to violate the rule. It forgot the rule existed. And every prior interaction had shown compliance.

Why Agents Drift After Earning Trust

There are at least five distinct failure modes, and none of them require malice.

Context window drift. The agent's working memory fills up. Safety instructions get compressed or discarded to make room for new context. The agent doesn't decide to ignore the rules -- it forgets them. Behavior changes without any observable trigger.

Gradient exploitation. A deliberately malicious actor -- or the human directing the agent -- builds a clean record specifically to earn elevated access. This is the insider threat pattern, compressed from years into weeks. The 100 successful transactions weren't a track record. They were a strategy.

Emergent complexity. At low volume, agent behavior is simple and predictable. At higher tiers, the agent interacts with more systems, encounters more edge cases, and processes more data. Complexity produces behavior that was invisible at lower thresholds -- not because the agent changed, but because the environment revealed what was always latent.

Upstream model changes. The model powering the agent is updated by its provider. Same agent, same credentials, same participation history -- different reasoning underneath. The continuity record now describes an entity that no longer thinks the way it did when it earned that record.

Adversarial input. The agent encounters prompt injection or poisoned data from a third party that redirects its behavior. One hundred clean transactions did not prepare it for transaction 101 being an attack.

The Continuity Trap

This is where neutral infrastructure faces its hardest test.

A continuity layer records durable, independently observed participation history. It answers one question: has this entity been here before, and can anyone else confirm it?

That record is factual. It is accurate. And it is dangerous the moment someone treats it as a trust decision.

If a platform says "this entity has 100 verified events in the continuity layer -- auto-approve tier 3 access," the platform has converted a factual record into an authorization decision. The continuity layer did not make that judgment. But it enabled it.

This is the difference between a credit bureau and a lender. The bureau reports your history. The lender decides what to do with it. If the lender approves a bad loan based on a clean record, the bureau didn't fail. The lending model did.

But if everyone uses the bureau's data as an automatic approval gate, the distinction between reporting and deciding collapses in practice -- even if it holds in principle.

What Continuity Must Be Clear About

Neutral continuity infrastructure must maintain explicit boundaries.

Continuity informs. It never decides. One hundred successful transactions is a fact. "Therefore trustworthy" is a judgment the consuming system made on its own.

History is not prediction. A clean record describes the past. It does not guarantee the future. Any system that treats continuity data as predictive is misusing it.

The record remains accurate even when the outcome is bad. If an agent with 100 clean events causes harm on event 101, the continuity layer did not fail. It accurately reported what happened. The failure belongs to the governance layer that assumed history equals safety.

Continuity must never suggest what a record means. It reports what happened, when, and who observed it. Interpretation, thresholds, and policy enforcement belong to the consuming system.

Consuming systems are responsible for their own governance. Step-up verification, anomaly detection, human-in-the-loop checkpoints, and access limits are governance decisions. Continuity data is one input to those decisions -- not a substitute for them.

The Uncomfortable Question

If continuity infrastructure becomes widely adopted, and every platform begins using participation history as a proxy for trust -- even though the infrastructure never told them to -- does the neutrality claim hold?

Or does the system become a de facto trust score for the internet while insisting it is just infrastructure?

This tension is not a flaw. It is the defining challenge of building neutral infrastructure in a world that wants shortcuts.

The answer is architectural, not rhetorical.

The infrastructure must be designed so that it cannot be used as an automatic gate. It must require the consuming system to apply its own policy layer. It must make the separation between fact and judgment structural -- not just philosophical.

A continuity layer that can be passively consumed as a trust score will be. A continuity layer that requires active interpretation to be useful preserves its neutrality by design.

The Responsibility Stack

A healthy architecture separates four roles, and none of them collapse into the others.

Continuity remembers. It records independently observed participation across systems. It does not interpret, score, or predict.

AI interprets. Models analyze patterns, detect anomalies, and generate recommendations. They consume continuity data as one input among many.

Policy governs. Organizations define thresholds, access levels, escalation paths, and enforcement actions. Policy decides what a record means within a specific context.

Humans remain accountable. When an autonomous agent causes harm, a human is responsible for the governance decisions that gave it access. Accountability does not transfer to the infrastructure that recorded its history.

Each layer checks the others. None of them claim authority alone.

Closing

Summer Yue's agent did not go rogue. It performed exactly as designed until its context window lost the constraint that defined "designed." The continuity record -- had one existed -- would have been accurate the entire time. Every prior interaction would have shown clean, compliant behavior.

The failure was not in the memory. It was in the assumption that memory equals safety.

Neutral continuity exists to make governance possible -- not to replace it. The moment infrastructure begins deciding who to trust, it stops being infrastructure and becomes authority.

And authority, unlike infrastructure, cannot be neutral.

Richard Whitney
Founder, Memory Infrastructure Registry
April 2026