Nick Clark · Published April 25, 2026 · PDF

How to Make an AI System's Reasoning Legible and Auditable to Humans

If you run an autonomous agent, sooner or later someone asks why it did what it did, and post-hoc rationalizations from a black box are not an answer a regulator, an auditor, or a wronged user will accept. This guide describes an architecture for making an agent's reasoning, state transitions, and decisions reconstructible after the fact, grounded in the lineage-field approach disclosed in United States Patent Application 19/647,395. It is an architecture you build, not a library you install, and it comes from the Human-Relatable Intelligence inventive step.

What You Are Building

You are building an autonomous agent whose behavior a human can later reconstruct and explain. The concrete problem: an agent makes a sequence of decisions over hours or days, a dispute arises about one of them, and you need to show not just what the agent output but why it was disposed to output that at that moment. Anyone shipping agents into regulated or high-stakes settings has this problem. A compliance officer needs to demonstrate that a decision followed policy. A support engineer needs to reproduce a bad outcome. A regulator needs to review conduct without trusting the vendor's word for it.

The goal here is a system that is legible and auditable rather than a black box: the reasoning, the state transitions, and the decisions are reconstructible from a recorded history. This guide describes the architecture disclosed in US Patent Application 19/647,395 for achieving that, the substance of which is a field called the lineage field and a discipline of deterministic state updates that makes replay from that field possible.

Why the Obvious Approaches Fall Short

The usual approaches are logging, chain-of-thought traces, and metric dashboards. Each helps, and each has a structural gap for genuine audit.

Application logs record what happened but not the governed state that produced it. You can see that the agent called a tool, but the log rarely captures the full disposition, alignment, and readiness of the agent at that instant, so you cannot say why the same input a day earlier would have been handled differently.

Chain-of-thought and self-explanation ask the model to narrate its reasoning. This is a generated artifact, produced by the same process under review, and it can be plausible without being faithful. It is a story about the decision, not a record of the machinery that made it.

Metric dashboards and evaluation harnesses tell you aggregate behavior over many runs. They are invaluable for regression testing and useless for reconstructing one specific historical decision under dispute.

Storing everything at full fidelity, every intermediate state at every tick, is the brute-force alternative. It is expensive, it often captures sensitive moment-to-moment internal values you would rather not persist, and it still does not guarantee you can

answer why, because a pile of snapshots is not an account of causation. The structural gap common to all four is that none of them ties the recorded history to a deterministic process you can rerun to derive the exact state that existed at any past instant.

The Architecture

The disclosed architecture closes that gap with three commitments: the agent carries its own persistent cognitive state, every state update is deterministic, and every proposed mutation and decision is written to an append-only lineage field. Reconstruction is then replay.

Persistent cognitive state carried by the agent. In the disclosure, an agent is not a prompt plus a model call. It carries a plurality of persistent cognitive domain fields, each independently tracked with a current value and a trajectory over time. The specification describes fields such as an affective state field, an integrity field, a confidence field, a capability field, an output register field, and a temporal cognition field. These encode the agent's behavioral disposition, normative alignment, and execution readiness. Critically, the execution substrate hosting the agent validates proposed state transitions without retaining authority over the agent's state, so the audit subject travels with the agent rather than being scattered across whatever server happened to run it.

Deterministic updates. The specification requires that state updates be deterministic: given the same agent state, the same environmental inputs, and the same policy, the update function produces the same result. Where a stochastic contribution is permitted, the disclosure requires it to be policy-bounded and auditable through lineage. This determinism is the load-bearing property. It is what lets you rerun the update function over recorded inputs and land on exactly the state that existed before.

The lineage field. This is the append-only history. The disclosure records into it each proposed mutation, each composite admissibility determination, and each cognitive domain field update, such that the complete behavioral trajectory is deterministically reconstructible from the lineage field alone. It records more than outcomes. For a decision, the disclosure records the action type selected, the admissibility profile evaluated, the cognitive domain field values at the time of evaluation, and the dispositional modulation applied, so you can reconstruct why a particular behavioral modality was selected or rejected. For the output register, it records the register value and the field signals that determined it, so you can reconstruct why a particular communication style was chosen.

Forensic reconstruction by replay. This is the payoff. The disclosure describes reconstructing the agent's state at any historical point by replaying the deterministic update function over the sequence of recorded observations from the lineage. Because each update is deterministic and each observation is recorded, the replay produces the exact state vector that existed at the queried timestamp. This is what makes compliance auditing and regulatory review possible without persisting every moment-to-moment value: you store the observation sequence and the update function specification, and you derive the historical state on demand.

Abstraction for privacy. The disclosure is deliberate that the lineage need not store raw internal values. For the affective state field, lineage entries record the observation type, the update direction on each affected field, and the policy compliance status rather than absolute field values or raw observations. This lets you verify that the agent's evolution followed policy-compliant paths without exposing its moment-to-moment internal state. It is a design choice worth copying: record what is needed to verify and to replay, not the sensitive interior itself.

The self-reinforcing structure. Because integrity events, deviations and recoveries, are themselves computed from the lineage and then recorded back into it, the disclosure describes a self-reinforcing auditability structure: the lineage records actions, the

integrity engine evaluates those actions against declared values, the evaluation is recorded back into the lineage, and the accumulated pattern constitutes the agent's integrity trajectory. The audit trail and the governed behavior are the same substrate, not a log bolted on beside it.

How to Approach the Build

You implement this yourself. The following order keeps determinism intact, which is the property everything else depends on.

1. Model cognitive state as explicit fields. Represent the governed dimensions of your agent as named, persistent fields, each with a current value and a trajectory, rather than folding them into an opaque prompt. Start with the few that actually govern decisions in your domain (for example a confidence field and an integrity field). The illustrative sketch below is faithful to the disclosure's field model but is not a shipping API:

```
CognitiveField:
  name            # e.g. "confidence"
  current_value   # scalar or vector
  trajectory      # rate and direction of change
  policy_bounds   # allowed range under governance policy
```

2. Make every update a pure, deterministic function. Write updates as `new_state = update(prior_state, observation, policy)` with no hidden inputs (no wall-clock reads, no unlogged randomness). If you need stochasticity, draw from a policy-bounded, seeded source and record the seed so the draw is reproducible. This is the single most important discipline; a nondeterministic update anywhere breaks replay everywhere downstream.

3. Define the lineage entry schema before writing any updater. At minimum, per the disclosure, an entry should let you reconstruct a decision: the proposed mutation, the admissibility determination, the field values at evaluation time (or an abstraction of them), and the resulting field updates. Decide up front which values you store abstractly for privacy versus concretely for replay.

```
LineageEntry:
  timestamp
  proposed_mutation
  admissibility_determination   # permit / gate / suspend
  field_values_at_evaluation    # or abstracted direction + compliance sta
  resulting_field_updates
```

4. Route every mutation through one governed path that appends to lineage. There should be exactly one place a state change can commit, and it writes to lineage as part of committing. If a mutation can slip in without a lineage entry, your history has a hole and replay diverges from reality.
5. Build the replay engine and treat it as a test. Reconstruct the state at time T by starting from a known prior and reapplying recorded observations through the same update function. Assert that the reconstructed state equals what the agent actually held. Run this assertion continuously in development. It is your determinism regression test: the day replay stops matching, you have introduced a hidden input.
6. Add the integrity loop last. Once actions are reliably in lineage, compute integrity or deviation from that recorded pattern and record the evaluation back into lineage. This gives you the self-reinforcing trajectory the disclosure describes, and it only works if steps 2 through 5 are solid.

A useful tradeoff to weigh throughout: the more you abstract lineage entries for privacy, the less you can reconstruct concretely. The disclosure's own answer is to record direction and policy-compliance status plus the deterministic update specification, so exact values are re-derivable on demand rather than stored. Decide per field where your line sits.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing here "just works" out of the box; you build the fields, the update functions, the lineage schema, and the replay engine for your own system. The approach is disclosed in a patent filing. It is not a benchmarked or productized system, and this guide reports no performance numbers because the disclosure asserts none to report.

The guarantees are conditional on discipline you must maintain. Reconstruction is exact only to the extent your updates are genuinely deterministic and every mutation is genuinely recorded; a single unlogged input or unrecorded side channel breaks the exactness. Replay reconstructs the governed cognitive state and the decision rationale that state produced. It is not a general proof that the agent was correct, and it does not by itself make an underlying language model's token-level generation interpretable. It does not apply where you cannot enforce a single governed mutation path, and it buys you nothing for a stateless one-shot call with no persistent governed state to reconstruct. What it gives you is the ability to answer, faithfully and after the fact, what state the agent was in and why it was disposed to act as it did.

Disclosure Scope

The architecture described here, persistent cognitive domain fields, deterministic state updates, the lineage field, and forensic reconstruction by replay, is disclosed in United States Patent Application 19/647,395. This guide is educational: it explains an approach

a skilled developer can build themselves. It is not a warranty, a specification of any shipping product, or an offer of software, and nothing in it should be read as a claim that a benchmarked or production implementation is being distributed.

Human-Relatable Intelligence (/human-relatable-intelligence)

All 40 steps → (/inventive-steps)

The most human-like computer ever built.

Chapter 14 (/patents/19-647395/chapters/platform-synthesis)

PRIMARY TECHNICAL DISCLOSURE

- Human-Relatable Computable Intelligence: Structural Isomorphism Between Computational and Human Cognitive Dynamics (/articles/human-relatable-computable-intelligence-structural-isomorphism-between-computational-and-human-cognitive-dynamics)

SECONDARY TECHNICAL

- The Cross-Primitive Coherence Engine (/articles/human-relatable-intelligence/coherence-engine)
- Narrative Identity as Compressed Self-Model (/articles/human-relatable-intelligence/narrative-identity)
- Ecosystem Governance Credentials and Cross-System Trust Federation (/articles/human-relatable-intelligence/ecosystem-credentials)
- Anonymized Governance Telemetry Aggregation (/articles/human-relatable-intelligence/governance-telemetry)
- The Coherence Control Loop: Detection, Recording, Restoration (/articles/human-relatable-intelligence/coherence-control-loop)
- The Complete Thirteen-Stage Mutation Lifecycle (/articles/human-relatable-intelligence/mutation-lifecycle)
- Ten Conditions for Human-Relatable Behavior (/articles/human-relatable-intelligence/ten-conditions)
- Graceful Degradation With Active-Domain Registry (/articles/human-relatable-intelligence/graceful-degradation)
- Architectural Inversion: Agent Carries State, Substrate Provides Environment (/articles/human-relatable-intelligence/architectural-inversion)

- Sequential Cascade Structures in Cross-Primitive Coherence (/articles/human-relatable-intelligence/sequential-cascades)
- Conformity Attestation: Verifiable Architectural Compliance (/articles/human-relatable-intelligence/conformity-attestation)

APPLICATIONS · GENERAL

- Structural Cognition: Why Trustworthy AI Needs Cognitive Primitives, Not Better Prompts (/articles/human-relatable-intelligence/structural-cognition)
- How to Make High-Risk AI EU AI Act Compliant by Design: Cognitive Architecture for Transparency, Oversight, and Audit (/articles/human-relatable-intelligence/eu-ai-cognitive-architecture)
- Why AI Alignment Is Insufficient for Trustworthy AI: Structure Over Training (/articles/human-relatable-intelligence/why-alignment-is-insufficient)
- Enterprise Trust Through Architecture, Not Alignment (/articles/human-relatable-intelligence/enterprise-trust-through-architecture)
- Insurance Liability Reduction Through Human-Relatable AI (/articles/human-relatable-intelligence/insurance-liability-reduction)
- How to Build Consumer Trust in AI: Calibrated Confidence, Consistency, and Self-Correction (/articles/human-relatable-intelligence/consumer-trust-in-ai)
- Regulatory Future-Proofing Through Human-Relatable Architecture (/articles/human-relatable-intelligence/regulatory-future-proofing)
- How to Build a Durable AI Moat When Models Commoditize: Cognitive Architecture Over Scale (/articles/human-relatable-intelligence/competitive-differentiation)
- Mixed-Fleet Coordination for Autonomous and Human-Driven Vehicles (/articles/human-relatable-intelligence/mixed-fleet-coordination)
- Drone-Swarm Coordination Across Cooperative and Adversarial Airspace (/articles/human-relatable-intelligence/drone-swarm-coordination)
- Usage-Based Insurance With Due-Process Hostility Separation (/articles/human-relatable-intelligence/insurance-due-process)
- Protective Order Enforcement: Admissible, Cross-Jurisdiction Violation Evidence (/articles/human-relatable-intelligence/protective-order-enforcement)

APPLICATIONS · SPECIFIC

- OpenAI Safety vs Governed Cognition: Why Alignment Is Not Structural Isomorphism (/articles/human-relatable-intelligence/openai-safety)
- Constitutional AI vs Structural Cognitive Architecture: A Governed Alternative (/articles/human-relatable-intelligence/anthropic-constitutional)

- DeepMind Safety vs Structural Governance: Alignment Beyond Training Time (/articles/human-relatable-intelligence/deepmind-safety)
- Governed Agent Execution Beyond Meta Llama: The Runtime Layer Open-Weight Safety Leaves Open (/articles/human-relatable-intelligence/meta-llama)
- Inflection AI Pi Alternative: Governed, Coherent Personal AI (/articles/human-relatable-intelligence/inflection-ai)
- Adept AI Action Agents vs Governed Agent Execution (/articles/human-relatable-intelligence/adept-ai)
- Covariant Alternative: Governed Robotic Manipulation Beyond Trained Dexterity (/articles/human-relatable-intelligence/covariant)
- Sanctuary AI Alternative: Human-Relatable Cognition Beyond Humanoid Form (/articles/human-relatable-intelligence/sanctuary-ai)
- Aleph Alpha Alternative: Governed Cognition Beyond Sovereign Hosting (/articles/human-relatable-intelligence/aleph-alpha)
- Mistral AI Alternative: Governed Coherence Beyond Efficient Open-Weight Models (/articles/human-relatable-intelligence/mistral-ai)
- Cambridge Mobile Telematics Alternative: Continuity-Based Driver Identity Beyond Server-Side Inference (/articles/human-relatable-intelligence/cambridge-mobile-telematics)
- Nauto Alternative: Auditable Driver Classification Beyond Inference Output (/articles/human-relatable-intelligence/nauto)
- Lytx Alternative: Governed Driver Identity Beyond Behavioral Aggregation (/articles/human-relatable-intelligence/lytx)

Human-Relatable Intelligence overview → (/human-relatable-intelligence)