Nick Clark · Published April 25, 2026 · PDF

How to Give an AI System's Decisions a Due-Process Record

You have an autonomous agent that makes consequential decisions, and now someone (a human owner, an insurer, a regulator, a court) wants to know why it decided what it did at a specific moment. Most systems cannot answer that question faithfully after the fact. This guide describes an architecture for making an agent's decisions reconstructible from an immutable lineage record, so the reason for any past decision can be reproduced rather than guessed. It teaches an architecture disclosed in United States Patent Application 19/647,395 (it is not a shipping library), grounded in the Human-Relatable Intelligence inventive step.

What You Are Building

You are building an agent whose past decisions can be examined the way a court examines a decision by a person or an institution: not by asking the agent to narrate its reasoning after the fact, but by reconstructing, from a preserved record, the exact inputs and rules that produced a specific decision at a specific time.

The problem shows up the moment an autonomous system does something consequential and someone with standing (the operator who is liable, the insurer underwriting the deployment, an auditor, a regulator, an opposing party in litigation) asks: why did it do that, and could it have done otherwise? "Due process" here is used

in its ordinary sense. A decision has a due-process record when there is a faithful, tamper-evident account of what was considered, what rule was applied, and why a given outcome followed, such that a neutral third party can check the account rather than take the operator's word for it.

This guide describes an architecture for that property. It is drawn from a patent filing, not a package you install. You implement it.

Why the Obvious Approaches Fall Short

The common approaches are honest attempts, and it is worth being precise about where they leave a gap rather than dismissing them.

Logging model outputs. Recording the prompt and the response tells you what happened, not why. The next run with the same input can differ, and the log contains no rule you can re-apply, so there is nothing to reconstruct against.

Asking the model to explain itself. A generated explanation is a fresh output, produced after the decision, conditioned on the request for an explanation. It can be plausible and still not be the actual cause of the earlier decision. This is a well-known limitation of post-hoc rationalization: the narrative is not causally tied to the original computation.

Storing full state snapshots at every step. Snapshotting the agent's entire internal state at every moment can be faithful, but it is expensive, and it forces you to persist moment-to-moment values (some of them sensitive) that you would rather not retain. It also does not, by itself, distinguish the governance decision from the values that happened to be present.

The structural gap common to all three: in a conventional agent stack, governance, memory, and decision eligibility live *outside* the agent, as an external scheduler, a vector store, a post-inference filter, or a system-prompt instruction. Because the

deciding logic is not part of a single recorded object, there is no one place that holds both the inputs to a decision and the rule that was applied to them. Reconstruction has nothing to anchor to.

The Architecture

The disclosed approach closes that gap by making governance and history *intrinsic typed fields of the agent object itself*, and by insisting that the record of decisions be complete enough to replay.

1. Put the decision-relevant state inside the agent. In the disclosed schema, the agent carries canonical fields (intent, context, memory, a policy reference, a mutation descriptor, and a lineage field) plus cognitive domain fields such as integrity, confidence, and capability. Each cognitive field is tracked as a current value and a trajectory over time. The point for auditability is that the state a decision depends on travels with the agent as structured, independently readable fields, not scattered across external services.

2. Route every change through a single admissibility gate. No field is mutated directly. A proposed change is expressed as a *proposed mutation*, and a *composite admissibility determination* evaluates it against policy, integrity, and readiness signals, producing one of a small set of outcomes: permit, gate, or suspend. The spec describes the admissibility gate as evaluating policy constraints, descriptor validation, lineage continuity, and entropy bounds to yield admit, reject, or defer. Because every decision passes through one evaluator, there is exactly one place where a decision is made and can be recorded.

3. Record the decision, not just the outcome, in an immutable lineage field. Each proposed mutation, each admissibility determination, and each resulting field update is written to the lineage field. Cognitive field changes are recorded in the same

lineage chain as every other state transition, so there is a single ordered history. The spec's governing requirement is explicit: the complete behavioral trajectory of the agent must be *deterministically reconstructible from the lineage field alone*.

4. Keep the deciding functions deterministic. Reconstruction only works if replaying recorded inputs reproduces the recorded result. The disclosed update functions are deterministic: given the same prior state, the same recorded inputs, and the same policy, they produce the same output. The spec notes that a stochastic contribution is permissible only if it is policy-bounded and itself recorded in lineage, so determinism of the replay is preserved.

5. Reconstruct on demand by replay. This is the payoff. Rather than persisting every intermediate value, you preserve two things: the lineage record of inputs and decisions, and the specification of the update functions. To answer "what was the agent's configuration at the time of the disputed decision," you replay the deterministic update function over the recorded sequence of observations from lineage up to the queried timestamp. Because each step is deterministic and each input was recorded, the replay yields the exact state that existed then. The spec describes precisely this for the affective state field: forensic reconstruction on demand from the lineage record and the update-function specification, without persisting moment-to-moment values.

6. Record the reason, not only the value. For decisions about which action was permitted, the lineage entry records the action type selected, the admissibility profile evaluated, the cognitive field values at evaluation time, and any dispositional modulation applied. The spec states this enables forensic reconstruction of *why* a particular behavioral modality was selected or rejected at any point in the interaction history. That is the difference between a log and a due-process record: the applied rule and the inputs to it are both present.

How to Approach the Build

The following steps are how a developer would implement the architecture themselves. The interface sketch below is illustrative and faithful to the spec; it is not shipped code.

Step 1: Define the agent object as the unit of record. Model the agent as a single object that owns its policy reference, its decision-relevant fields, and its lineage. Resist the instinct to keep governance in a separate service; the whole property depends on inputs and rules being co-located with history.

Step 2: Make mutation the only way state changes. Every change is a proposed mutation with a descriptor. Nothing writes a field directly.

```
# illustrative interface sketch, not a library
proposed = Mutation(target_field, proposed_value, action_type, observations)
decision = admissibility_gate.evaluate(proposed, agent.fields, agent.policy_
# decision.outcome in {ADMIT, REJECT, DEFER}
lineage.append(LineageEntry(
    proposed = proposed,
    outcome = decision.outcome,
    fields_at_evaluation = snapshot_of_inputs_only(proposed, agent.fields),
    profile_evaluated = decision.admissibility_profile,
    modulation_applied = decision.dispositional_modulation,
    timestamp = now(),
))
if decision.outcome == ADMIT:
    apply(proposed)   # deterministic update function
```

Step 3: Implement the admissibility gate as a pure function of recorded inputs. Its output must depend only on things you write to lineage: the proposed mutation, the relevant field values, and the policy. If the gate consults anything you do not record, reconstruction breaks.

Step 4: Make lineage append-only and integrity-protected. The spec ties lineage to the agent's cryptographic provenance and speaks of lineage continuity being validated. Treat the record as append-only and chain entries so that tampering is detectable; that is what lets a third party trust a replay rather than trust you.

Step 5: Record inputs and the applied rule, not derived values you can recompute. Store observations, the action type, the profile evaluated, and the modulation applied. Do not persist moment-to-moment derived state you can reconstruct by replay. The spec makes this an explicit design property: lineage records reference mutations abstractly and omit absolute values and raw sensitive signals, which both bounds storage and reduces what you retain about people.

Step 6: Build the reconstruction path as a first-class operation. Given a timestamp, load lineage up to that point, replay the deterministic update functions over the recorded observations, and return the state and the decision record. Test it by asserting that a replay reproduces the outcome recorded live at that step. If it does not, either a non-recorded input leaked into a decision or an update function is non-deterministic; both are bugs against this architecture.

Step 7: Handle suspension and preserve it too. When admissibility indicates insufficient readiness, the agent transitions to a non-executing cognitive mode instead of acting, and that transition is itself recorded. A due-process record must show the decisions *not* to act as clearly as the decisions to act.

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." You implement the agent object, the admissibility gate, the lineage store, the deterministic update functions, and the reconstruction path yourself, and you own their correctness.

It is not benchmarked or productized here. The filing discloses the method; this guide does not present performance numbers, and you should measure storage, replay cost, and latency in your own deployment.

The property is only as strong as your discipline. If any decision depends on an input you did not record, reconstruction will diverge from what actually happened. If an update function is non-deterministic in an unrecorded way, replay is invalid. If lineage is not integrity-protected, a third party has no reason to trust the replay. The architecture makes faithful reconstruction *possible*; it does not enforce it against a sloppy implementation.

Finally, this addresses reconstructing the agent's own governed decisions. It does not adjudicate whether the underlying policy was wise or lawful, and it does not reconstruct behavior of external systems the agent merely called. It gives you a faithful record of what your agent decided and why, which is the input to that larger judgment, not a substitute for it.

Disclosure Scope

The architecture described here is disclosed in United States Patent Application 19/647,395. This guide is educational: it explains the disclosed approach so a skilled developer can understand and build it, and it is grounded solely in that filing. It is not a warranty, a specification of any product, or an offer of software, and it does not grant any license under the referenced application. Any implementation, and its fitness for a given governance, insurance, or legal purpose, is the reader's responsibility.

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)