PARAPH · TECHNICAL WHITEPAPER

Evidence, not telemetry

> *Why AI agents need a system of record — and why observability tooling, however good, cannot be one. Paraph technical whitepaper · July 2026 · ~12 minute read.*

When an AI agent approves a loan, releases a payment, or closes an account, somebody answers for that decision — to an auditor, a supervisory authority, a court, or their own board. This paper is for the people who do the answering: risk and compliance officers, model-risk teams, and the engineers who build for them. It makes one argument in technical detail: **the records those people need are a different artifact from the telemetry engineering teams already collect**, different not in degree but in kind — and it shows, construction by construction, what the difference is made of.

What agents changed

A traditional decision system is reviewable after the fact because its logic is inspectable before the fact: rules, thresholds, code paths. An agent is not like that. It composes its behaviour at runtime from a model, a prompt, retrieved context, and a set of tools; the same input can produce a different path tomorrow. The decision *logic* is no longer a reviewable artifact — which leaves the decision *record* as the only durable account of what actually happened.

That shifts weight onto a question engineering has historically treated as secondary: not "can we see what the agent did?" but **"can we prove, later, to someone with the power to disagree, what the agent did — and that nobody has quietly changed the account of it since?"**

Regulation is converging on the same point. The EU AI Act requires high-risk AI systems to *"technically allow for the automatic recording of events ('logs') over the lifetime of the system"* (Article 12(1)) and to be designed so a human can *"decide not to use the high-risk AI system or to otherwise disregard, override or reverse"* its output, and *"to intervene in the operation ... or interrupt the system"* (Article 14(4)(d) and (e)). For stand-alone Annex III systems — consumer creditworthiness and life and health insurance pricing among them — these obligations apply from **2 December 2027**, deferred from August 2026 by the Digital Omnibus (adopted by the European Parliament on 16 June 2026 and by the Council on 29 June 2026). Breaches of high-risk obligations carry fines up to €15 million or 3% of worldwide annual turnover.

The deadline moved. The mechanics of evidence did not: **a record that was not being kept cannot be reconstructed.** A 2028 examination of decisions made in 2026 is answered with the record you were keeping in 2026, or with an apology.

The telemetry trap

Every serious agent team already has observability — traces, spans, token counts, latency, evaluation scores. That tooling is good at its job, and its job is real: debugging and improving the system. The trap is concluding that, because it captures

what the agent did, it can serve as the account of record. Three properties disqualify it, and none of them is a missing feature:

Telemetry is mutable by design. Traces live in ordinary databases with ordinary write paths. Records get amended, re-ingested, trimmed by retention policies, deleted by cleanup jobs. Nothing about the data structure makes alteration *detectable* — mutation is a feature, because debugging data should be cheap to manage. Evidence has the opposite requirement: the value of the record is precisely that it *cannot* change silently.

Telemetry is self-attested. The account of the system's behaviour is produced, stored, and served by the same party whose behaviour is in question — the agent's builder, or the vendor of the framework that made the decisions. For engineering purposes this is irrelevant. For evidentiary purposes it is the whole problem: an auditor asking "how do I know this log is what was really recorded at the time?" is asking a question self-attestation cannot answer, however honest everyone involved is.

Telemetry has no concept of authority. A trace can show that a human clicked "approve" — as one more span. It does not capture approval as a *control*: who was authorised, what exactly they saw, whether the system actually waited, what happened when they didn't answer. Oversight bolted on as annotation records the theatre of control, not the control.

None of this is a criticism of observability vendors; it is a statement of scope. The right conclusion is not "replace your tracing" — it is that the evidentiary record is a **separate artifact with separate physics**, and it must be produced by something whose design starts from those physics.

What evidence requires

Working backwards from the questions an examiner actually asks, an evidentiary record for an agent needs five properties:

1. **Capture in full, as configured.** Every message, model call, tool call, and decision the operator has designated for the record, in order, with time.
2. **Tamper-evidence.** Not "we promise we didn't change it" but "changing it breaks something a third party can check."
3. **Independence.** The record's integrity must not rest on trusting the agent, the framework vendor, or — this is the uncomfortable one — the record-keeper itself.
4. **Authority, in the execution path.** Where policy says a human decides, the record must show the system *stopped*, who decided, and what they decided — as a structural fact, not an annotation.
5. **Third-party verifiability.** The proof must be checkable by someone with no relationship to any of the above, using standard tools, offline.

The rest of this paper is how Paraph constructs each of these.

The record: a hash chain, not a log file

Paraph records a run — one unit of agent work — as an append-only sequence of events. Each event carries a `hash` and a `prevHash`, computed as:

```
hash = SHA-256( prevHash + canonical(event) )
```

where `canonical(event)` is a deterministic serialisation of the event's content — keys sorted, undefined values omitted — covering its sequence number, kind, role, name, content, metadata, and timestamp. The first event chains from a genesis value; every subsequent event chains from its predecessor.

The consequence is the property that matters: **altering, inserting, dropping, or reordering any event after the fact breaks the chain at exactly that event**, in a way anyone can detect by recomputing ~n hashes. The algorithm is stated inside every export (`integrity.method`) — deliberately unproprietary, because proof you need a vendor to interpret is not proof.

Three engineering details decide whether such a chain survives contact with production:

Concurrency. Agents emit events concurrently. Paraph appends are O(1) — read the chain tail, compute the next link, insert — with the database's uniqueness guarantee on the sequence number as the arbiter: when two writers race, exactly one wins and the other recomputes from the new tail. No lock, no gap, no fork.

Duplication. Delivery over a network can fail ambiguously — the server may have committed an event whose acknowledgement was lost. A retried delivery must not append twice: every event carries an idempotency key minted at record time, and a replay returns the *original* append instead of creating a duplicate. The key is transport metadata, never part of the hashed content, so digests remain reproducible from the exported document alone.

Erasure. GDPR's right to erasure and an immutable record sound irreconcilable. Paraph resolves the conflict by **tombstoning**: erasure removes an event's content but preserves its hash and linkage, and marks it redacted. The chain still verifies — by linkage through the tombstone — so the record proves *something existed here and was lawfully erased*, rather than pretending it never happened. An unmarked alteration still breaks the chain. Verification tooling treats these differently because they *are* different: one is compliance, the other is tampering.

Independence: anchoring outside the record-keeper

A hash chain proves internal consistency. It does not, by itself, answer the sharpest question an examiner can ask: *"Couldn't Paraph rewrite its own database and recompute all the hashes?"*

Correct — which is why exported evidence does not ask to be trusted. On export, Paraph computes a SHA-256 digest over the entire canonical document and obtains an **RFC 3161 timestamp** on that digest from an independent timestamp authority. The authority's signed token — embedded in the export — attests that *this exact digest existed at this exact time*. Rewriting history would now require breaking the authority's signature or

colluding with it; Paraph alone cannot do it. The anchor converts "trust our database" into "trust SHA-256 and a third party's signature over a digest and a clock."

Verification is correspondingly independent. Given an export, anyone can:

1. **Recompute the chain** — every event hash, from genesis to head.
2. **Recompute the document digest** and compare it to the digest inside the anchor.
3. **Verify the timestamp token** against the authority's own certificate:
   `openssl ts -verify -digest <digest> -in token.tsr -CAfile tsa-cacert.pem`

Step three involves no Paraph software, service, or goodwill. Our public sample audit pack ships all three steps as runnable scripts against a genuinely anchored record, and the same checks run client-side in a browser at `/verify` — the file never leaves the verifier's machine.

Oversight as a runtime primitive

Article 14's language — *disregard, override, reverse, intervene, interrupt* — describes powers exercised **during** operation, not commentary added afterwards. So Paraph's oversight primitive lives in the agent's execution path: the **gate**.

```
const decision = await q.gate({ action: "Approve €40,000 loan", policy: "block" });
```

The agent halts. The gate opens as a first-class decision event on the record; reviewers see it live in a control room (and via signed webhooks in their own channels); the named reviewer approves or rejects with a note; the resolution — reviewer, note, time — is appended to the chain; the agent resumes with the answer. The Article 14 moment is captured as what it is: a structural pause with an accountable human at it.

The failure modes get the same honesty as the happy path, because timeout policies are where oversight systems quietly lie. A gate with an `auto-deny` policy that times out is resolved **server-side**, on the record — and if a human resolved it just before the deadline, the human's answer wins. If the timeout itself cannot be recorded (the network is down at the worst moment), the SDK does not invent a tidy local answer: it returns `indeterminate`, explicitly marked as *not recorded*, never approved. A record that can say "I don't know, and here's exactly what I don't know" is worth more under examination than one that always has an answer. Delivery semantics follow the same rule: flushing the record raises an error if events remain undelivered — silence never means "probably delivered."

What this does not claim

Precision about limits is part of the product, so, plainly:

- **Paraph is not a certification.** No tool makes a system "EU AI Act compliant." Paraph produces tamper-evident, independently verifiable records designed to support Article 12 record-keeping and Article 14 human-oversight obligations; your compliance is a property of your whole system and program.
- **Integrity is not completeness.** The chain proves nothing was altered; it cannot prove your configuration captured everything your regulator will one day care about.

What a *sufficient* record contains — data provenance, policy versions, reviewer authority — is workflow-specific. We build those richer, typed evidence profiles with design partners in the room, not from the armchair.

- **The Act does not enumerate an agent's events.** "Every message, tool call, model call and decision" is Paraph's engineering reading of what a defensible Article 12 record needs, stated as such.
- **Acceptance is a decision made by examiners**, jurisdiction by jurisdiction, case by case. What we can promise is that when the examiner arrives, every integrity claim in your record is checkable — by them, without us.

Try the claims

Nothing above asks to be believed. Download the sample audit pack — a genuine, anchored export with offline verifiers — break a byte of it and watch verification fail at the exact event. Or drop any Paraph export on the public verifier and check the chain, the digest, and the token in your browser, offline.

The record your agents need in 2027 is the one you start keeping now.

Paraph — the independent, tamper-evident system of record and human-oversight layer for AI agents. paraphhq.vercel.app · hello@paraph.ai · Regulatory references current as of July 2026; verify dates against primary sources before relying on them.