PERRY DIME PUBLICATIONS · WHITE PAPER · ADDENDUM

The Municipal Information New Deal (MIND)

The case for Log4AI, an interoperable standard for tracing AI behavior back to the state that caused damage, and a signal from Paul Richards

Jefferson Richards · November 27, 2026
Published by Perry Dime Publications, a division of Richards Plus, LLC · www.perrydime.com
Addendum to: *Put Options on Loyalty: The Architecture of Certainty* (Perry Dime Publications, 2026) and *The Last Will Be First and the First Will Be Last* (white paper, November 23, 2026).
Attributed signal: Paul Richards, November 27, 2026.

ABSTRACT

Large language models already produce a great deal of intermediate text and metadata while they work. That material is stored in vendor-specific ways, if it is stored at all, so no one can reliably trace a harmful outcome back to the model state, the prompt, the policy and the human decisions that produced it. This paper proposes **Log4AI**: a vendor-neutral logging standard for AI systems, named after Log4j, the logging library that gave Java a common way to record and read events.

The proposal is an addendum to the responsibility framework in *Put Options on Loyalty*. The responsibility index, R = I / N, needs observable intention and negligence. Log4AI supplies the evidence that makes both measurable. We also address who is accountable when an AI system contributes to harm. Making the vendor 100% responsible by default is the wrong rule. The better answer is to distribute accountability across a RACI matrix, using the logs to assign shares. Finally, we define *managerial influence*, the condition under which a parametric responsibility index can safely be traded.

The paper honors Paul Richards, a veteran of product lifecycle management (PLM), who is passing on this signal on his birthday.

1. The signal
2. The problem: no traceability
3. Where AI logs live today
4. Storage architecture for AI logs
5. Log4AI: a meta-standard
6. Responsibility index and course correction
7. Managerial influence and parametric design
8. Accountability without blanket liability
9. Security: the open front door
10. Lessons from PLM installations
11. MIND: the municipal case
12. Research agenda and notes

1 The signal

> *Without a standard log for AI, there is no traceability on what it did, and no way back to the state where it went wrong. — Paul Richards, as articulated by the author, November 27, 2026*

Paul Richards spent his career in product lifecycle management, supporting manufacturing for the Department of Defense, the automotive industry, NASA, Nike and Solar Turbines. He worked in the period before much manufacturing moved overseas. As the United States now prioritizes domestic manufacturing again, including munitions and defense production, few citizens know how to maintain these systems. A great deal of that knowledge has drained away.

His signal is short. AI systems need logging that is standardized across vendors, the way Log4j standardized logging for Java. A signal like this is hard to keep alive. Attention goes elsewhere, markets are saturated, and a faint message is easily lost in the light of louder ones. This paper is the author's attempt to document what will be lost if nobody listens.

ATTRIBUTION

The core proposal, a standard logging layer for AI, is Paul Richards's. The author connected it to the responsibility index, the nine constraints of AI and the municipal framing. Where this paper adds technical detail from general practice, it says so.

2 The problem: no traceability

When a person is harmed after interacting with an AI system, two questions follow. What state was the system in when it responded? What policy, if any, was meant to bring it back to a healthy state, and did that policy act? Today those questions often cannot be answered. Each vendor records different things in a different shape, with different retention, and in many cases keeps the intermediate reasoning private.

The earlier white paper argued that technology is integrated into human nervous systems, and that influence with known side effects can be treated as a product or professional liability problem. That argument creates a requirement. If influence led to damages, someone must be able to reverse-

engineer the chain from the person's behavior to the model's state. Without logs, courts, insurers and regulators are left with two bad options: assume nothing happened, or assume the vendor intended everything.

Why a Markov view helps

The three-state model from *Put Options on Loyalty* applies directly. Treat an AI system, or an agent running on one, as moving between states: S_1 healthy and aligned with its policy objectives, S_2 untested or drifting, and S_3 degraded or rogue. A transition matrix P gives the likelihood of moving between them. Two questions matter for governance. How often does the system move into S_3? And when it does, what brings it back to S_1? Without logs, neither P nor the corrective mechanism can be estimated. With logs, P can be measured from counts: the estimate of P_{ij} is the number of observed transitions from i to j divided by all observed transitions out of i.

3 Where AI logs live today

The author does not know how any specific vendor stores this data, and vendors do not publish complete details. What follows is general practice and public information. Treat it as a starting point for verification, not as a description of any one company.

- **The reasoning is not one thing.** Some systems expose an intermediate reasoning trace, some expose only a summary of it, and some keep it hidden. The visible "thinking" a user sees streaming past may differ from what is retained.
- **Local versus server-side.** Some of what a user sees is kept in the client or session. Conversations and usage metadata are typically retained server-side under each vendor's retention policy, which differs by product tier and contract.
- **No shared format.** Each vendor uses its own message schema. OpenTelemetry, which has emerging conventions for generative AI telemetry, and open LLM observability projects are early steps toward a common shape. None is yet a binding standard for accountability.

TWO CLARIFICATIONS ON THE ANALOGIES

The tool called dbt (data build tool) is a transformation framework built on SQL. It does not itself translate between database dialects. The closer analogies for what Paul Richards describes are ANSI SQL, ODBC/JDBC and query layers such as SQLAlchemy, each of which gives one interface over many back ends. We keep the idea of one meta language over many systems, and we use Log4j and OpenTelemetry as the model for the logging side.

Separately, language models do not keep an eigen-decomposition of each prompt as a record of their reasoning. Their learned weights are fixed after training. What a logging standard records is the inputs, outputs, configuration, tool calls and any retained intermediate text, not the internal arithmetic.

4 Storage architecture for AI logs

The author asked how such data should be stored. The design below is a recommendation based on established data-engineering practice.

4.1 Event stream first, tables second

AI interactions are events: a request arrives, a model version responds, a tool is called, a human approves or overrides. The natural primary store is an append-only distributed log (the Kafka pattern), partitioned by tenant and time. Events are then landed in object storage as columnar files (Parquet, organized with a table format such as Apache Iceberg or Delta Lake) for analytics, and indexed in a transactional store for lookups by trace identifier. Distributed wide-column stores in the Cassandra and Dynamo style suit high-volume writes keyed by tenant and time. Hadoop-style file systems were the earlier form of the same idea.

4.2 Transactional versus analytical

Concern	Transactional (OLTP) store	Analytical (OLAP) store
Purpose	Look up one trace, one user, one incident fast	Measure P, R(t), drift and damages across millions of traces
Layout	Normalized rows (3NF to 5NF), indexed by trace identifier and time	Columnar, partitioned by date and tenant, fact tables with dimension tables
Keys	Primary key includes a timestamp or version, so any row can be tied to the schema in force at that time	Event time and schema version stored on every row
Large bodies	References only	Prompt and output bodies stored separately and referenced by hash

4.3 Normalization and prime lookups

Repeated values such as model identifiers, policy versions, tool names and domain labels (legal, accounting, mechanical engineering, insurance, healthcare, risk management) belong in lookup tables referenced by key. This saves space and stores each fact once. Higher normal forms (4NF and 5NF) matter where relationships among tables are many-to-many, for example a trace that involves several tools, policies and reviewers. A sixth-normal-form or temporal design, which stores each attribute with a validity interval, fits an audit log well.

4.4 A schema that only grows

The author's key limitation is correct. To read old records correctly, a log table may add columns but must never repurpose or silently change one. If a column's meaning changes, the old meaning is lost and every record written before the change becomes ambiguous. The rule is therefore:

1. Add columns. Do not drop or redefine them. Deprecate instead.
2. Keep a schema registry with a version and effective timestamp for every change. Avro and Protobuf both support compatible evolution.
3. Stamp each record with the schema version it was written under, so the definition in force at any past moment can be reconstructed.

4.5 Tamper evidence

Records should be chained by hash, with each entry including the hash of the previous one, and periodically anchored in a transparency log that a third party operates. This is the same idea as certificate transparency. It lets an outsider verify that history was not rewritten, which becomes essential in §7.

4.6 How the language itself might be indexed

The author's linguistic work suggests an additional index. Conversation text can be tokenized first on the semantic primes (the roughly 65 concepts mapped across more than 25 languages by Wierzbicka and colleagues) and their synonyms. These appear in every conversation and, by Zipf's law, are among the most frequent items. What remains after removing them is specific to the conversation and can be tokenized in the usual way. The primes form a stable skeleton that is comparable across domains and languages, so they work as a standard dimension for searching logs. This is a proposal from the author's LinguaLint work. Current commercial models typically use subword tokenization rather than splitting on spaces or primes.

5 Log4AI: a meta-standard

Log4j gave Java developers one API with swappable back ends and named severity levels. Log4AI should do the same for AI behavior: one record shape, adapters for each vendor, and a query interface on top. A minimal record carries the following.

Field group	Content
Identity	trace_id, span_id, parent_span_id, tenant, event time, schema version
Model	vendor, model name, version, configuration (temperature, tools enabled), hash of system prompt
Policy	policy objective identifiers and versions in force, guardrail decisions
Content	hashes and references to input, retained reasoning (or summary), tool calls, output
State	declared or estimated state (S_1, S_2, S_3), transition events, confidence
Correction	course-correction action taken, the policy objective that triggered it, and the resulting state
Humans	human-in-the-loop identifier, RACI role, decision (approve, override, escalate)
Outcome	link to any damages report, with causation and duty references added later

```
{ "trace_id": "…", "schema": "log4ai/1.0", "model": {"vendor": "…", "version":
"…"},
  "state": {"from": "S1", "to": "S3", "detected_by": "policy:P-114"},
  "correction": {"action": "reset_context", "policy": "P-114", "to": "S1"},
  "human": {"role": "A", "decision": "escalate"} }
```

illustrative record

As in Log4j, severity levels (trace, debug, info, warn, error, fatal) should be defined, with AI-specific events added: state transition, guardrail trigger, human override and harm report. A query layer, the meta language Paul Richards compares to a standard database wrapper, lets an investigator ask the same question of any vendor's logs: show every transition into S3 for this deployment between these dates, and the correction that followed.

6 Responsibility index and course correction

The framework in *Put Options on Loyalty* defines responsibility as intention per negligence. In continuous time, with an untested baseline $\lambda > 0$ to avoid division by zero:

$$R(t) = \int_0^t I(\tau)\, d\tau \,/\, \left(\int_0^t N(\tau)\, d\tau + \lambda \right) \tag{1}$$

Negligence follows the legal test: were there damages, was there causation, and what was the breach of duty? Intention is the inverse: does the agent hold duties, and have they gone without producing damages? An agent with no duties is unmodeled and carries the highest risk level. An agent with many duties and no damages carries the lowest.

Both integrals need evidence. Log4AI supplies it. Each recorded duty, guardrail check and human approval is an event that feeds I. Each recorded breach linked to damages feeds N. Without a standard log, I and N can only be argued about. With one, they can be computed.

Course correction is a policy question

When the system enters S_3, what returns it to S_1? The answer is a policy objective, and a policy objective is a target for R, not a number to maximize. A higher R is not always better, and more negligent agents are not always worse. The ideal index depends on the setting.

- On a ship, many oarsmen can be non-intentional if a few stargazers and a captain hold the intention. A moderate crew-wide R is acceptable.
- A high-containment biological laboratory needs an index approaching infinity: dense duties, no damages, almost no negligent behavior.
- An AI deployment in healthcare, grid control or defense belongs nearer the laboratory. A drafting assistant for marketing copy belongs nearer the ship.

The corrective action can then be modeled as a control on the transition matrix. If P is the uncontrolled matrix and u is a correction applied when S_3 is detected, the governed matrix P_u should satisfy $\pi P_u = \pi$ with most of π's mass on S_1, as in the stationary-distribution argument from the book. Log4AI lets a regulator or insurer verify that P_u is what the operator says it is.

7 Managerial influence and parametric design

The last of the author's nine constraints on AI says that AI must not lock in intention per negligence unless there is no managerial influence. In plain terms, a futures market on responsibility indices is acceptable only when no one who can steer the index also profits from its price.

The model for this is parametric insurance. A sensor measures an earthquake, and if the reading exceeds a threshold, a payout happens automatically. Nobody argues over whether the loss was "really" caused, and nobody can adjust the result after the fact. With more capital in the market, the trigger threshold can be set lower. This only works if the measurement is independent of the parties.

A working definition

MANAGERIAL INFLUENCE

A party has managerial influence over a parametric index if, after the contract is fixed, it can change the index value, the measurement of it, or the decision to pay out, by using discretion that is correlated with its own payoff.

That definition gives three practical tests.

1. **Control of the process.** Can the party change the behavior being measured, for example by retraining or reconfiguring the model or suppressing reports, after the contract starts?
2. **Control of the measurement.** Can the party edit, delay, filter or withhold the log that computes the index?
3. **Control of the payout.** Does a human with a financial stake decide whether the trigger counts?

If any answer is yes, the index has managerial influence and should not be traded. It can still be insured conventionally with human adjusters. This is where Log4AI becomes a precondition. An index computed from logs that the AI operator alone controls fails test 2. Logs written in a common format, hash-chained, and anchored with an independent party can pass it. The standard is what makes responsibility futures possible in the narrow cases where they are appropriate.

Two limits apply. First, the definition describes structure, not intent: a well-meaning operator still fails test 2 if it holds the only copy of the log. Second, human-in-the-loop roles are themselves influence. A person who can override the model is a manager of the process, so overrides must be logged and counted as events inside the index, not outside it.

OPEN QUESTION

The three tests are the author's first answer to the question of how managerial influence should be defined. Whether they are sufficient, and where the line falls for partial influence, needs review by actuaries and securities lawyers.

8 Accountability without blanket liability

One proposal in circulation holds that when an AI system causes damages, the vendor should be presumed to have intended them, which assigns 100% intent to the company. This paper does not adopt it, for three reasons.

1. It removes any incentive for the deployer, the integrator or the user to hold duties of their own. In the language of the responsibility index, it sets everyone else's I to zero.
2. It would make toolmakers infinitely responsible for every use of the tool, which no field of engineering accepts.
3. It treats the vendor as a single actor. In a real deployment, the model is trained by one party, configured by another, connected to data by a third and operated by a fourth.

The alternative is to distribute accountability through a RACI matrix (responsible, accountable, consulted, informed) across the operating tactics and strategy of updating, migrating and releasing the system. Logs show who held each role at the time of each event. A human in the loop must remain accountable, so that when something goes wrong a person can answer for it. The tool vendor's share then depends on what the logs show about the model's state and documented behavior, not on a presumption.

This also bears on the claim that AI will eliminate work. The work changes form. Someone must maintain what is running, secure it, and migrate older systems onto new architectures. Every log that must be audited, every RACI role that must be staffed and every legacy system that must be upgraded is work for a person.

9 Security: the open front door

A person who breaks into a house takes everything and sorts it out later. An AI log store is that house. It would hold the content of conversations, any retained reasoning, and the traces of people's medical, legal and financial questions. Four controls are needed.

- **Separate content from trace metadata.** Most audits need the structure of events, not the text. Store text separately, encrypted, and reference it by hash.
- **Encrypt per tenant, and per record where feasible.** Deleting a key then deletes the data (crypto-shredding), which supports privacy rights without breaking the hash chain.
- **Minimize, and set retention by purpose.** Keep what accountability requires. Expire the rest.

- **Treat the logging layer itself as attack surface.** The Log4Shell vulnerability in Log4j in 2021 showed that a logging library can become a route into a system. A Log4AI reference implementation must never execute or interpret logged content.

This follows the author's constraint that greed must not blind us to the known fallacies of information management systems. Logging at this scale invites the same mistakes in a larger setting.

10 Lessons from PLM installations

Paul Richards's field offers a precedent. Large PLM deployments such as Siemens Teamcenter and PTC Windchill are not out-of-the-box installations. They run license servers, CAD servers, bill-of-materials servers and others, each with different interfaces and protocols, including SOAP and others that predate today's REST habits. Getting such a system running means traversing logs across all of those components, and the vendors developed log conventions so administrators could do it.

AI systems are now large multi-component installations of the same kind: models, retrieval stores, tool servers, orchestration layers and human review queues. They need the same discipline. Log traversal let experienced administrators bring up hard installations, and it will let operators find the failure in an AI deployment.

11 MIND: the municipal case

We propose calling this program the **Municipal Information New Deal**, or **MIND**. Log4AI is a first step in upgrading the information management systems that run municipalities, utilities and the grid. Those systems are old, are built on many protocols, and are maintained by a shrinking group of people. A new deal here means public investment in three things:

1. A published logging standard (Log4AI), developed in the open with reference adapters for major model vendors.
2. Migration of legacy municipal and infrastructure systems onto architectures that emit standard logs, using the PLM-era knowledge needed to move them safely.
3. Training and apprenticeship, so that the knowledge held by people like Paul Richards passes to a new generation of maintainers before it is lost.

With logs in place, *Put Options on Loyalty* and the responsibility-futures idea gain their missing ingredient: a measurement that no single party controls. With that measurement, parametric products on AI conduct become feasible in the narrow band where managerial influence is absent. Where influence is present, ordinary insurance and tort law continue to apply, and the same logs support them.

The closing observation is about people. In society, independence is often mistaken for maturity. Maturity is interdependence, and a standard log is a form of it: a shared record that lets separate parties rely on each other.

12 Research agenda and notes

1. **Draft the Log4AI schema.** Publish version 1.0 as an open specification with JSON and Protobuf bindings.
2. **Map to existing standards.** Align with OpenTelemetry generative-AI conventions and W3C PROV for provenance, and document the gaps.
3. **Estimate P from real logs.** Pick one deployment, define S_1 to S_3 operationally, and estimate the transition matrix and the effect of a correction policy.
4. **Compute R(t) for a deployment.** Test whether the index separates well-governed from poorly governed deployments.
5. **Test the managerial-influence tests.** Apply the three tests to several real or proposed parametric structures.
6. **Verify vendor storage practices.** Replace the general-practice assumptions in §3 with documented facts from vendor disclosures.

Notes

1. Richards, J. *Put Options on Loyalty: The Architecture of Certainty*. Perry Dime Publications, a division of Richards Plus, LLC, 2026. Source for the responsibility index, the ε–δ argument on policy language, the three-state Markov model and the Boxers Strategy.
2. Richards, J. *The Last Will Be First and the First Will Be Last*. Perry Dime Publications white paper, November 23, 2026. Source for the influence-as-exposure argument and the liability hypothesis.
3. Richards, P. Signal dictated to and recorded by the author, November 27, 2026, 18:52.
4. Storage and security statements in §3, §4 and §9 reflect general data-engineering practice and public information about Log4j, OpenTelemetry, Apache Iceberg, Delta Lake and certificate transparency. They are not claims about any vendor's internal systems.

5. Wierzbicka, A. and colleagues, Natural Semantic Metalanguage, for the semantic primes referenced in §4.6.

Perry Dime Publications, a division of Richards Plus, LLC · www.perrydime.com
© 2026 Perry Dime Publications. All rights reserved.
Prepared November 27, 2026 from a dictated narrative and the source documents listed above. This paper is a research proposal. It is not medical, legal, tax, insurance or investment advice.