

Prove what the system knew

Six months after the incident, what can you actually show?

AT A GLANCE

The problem. The hardest part of an incident review is rarely the physics. It is reconstructing what the machine knew at the moment it decided, from logs written to help a developer, not to answer an investigator.

What this does. Accounts for every input that reached the system's recorded ingestion boundary — used, refused, or arrived late — records separately where nothing was received from an expected source, and produces a record that replays deterministically under stated conditions.

Who adopts it. The operator or integrator, who fits it and holds the record. **Who reads it:** casualty investigators, insurers, regulators and class societies, and counsel on either side — all of whom will ask how far it can be relied on.

Maturity. Implemented in a company-funded reference implementation and exercised in simulation. **Not yet proven in the field:** no field trial, no installed base.

Status. Covered by US provisional patent application **64/159,476**, filed 21 September 2026. One of four applications filed 2026; the fourth, on consensus over categorical judgements, has its own paper. Patent pending; none examined or granted.

The question everybody asks and almost nobody can answer

An automated system did something. Later — sometimes much later — four people want to know why: the investigator wants the sequence, the regulator the basis, the insurer whether the operator was negligent, counsel what was knowable at the time. One question in four registers: **what did the system know at the moment it decided?**

The usual answer is a reconstruction: logs pulled from several subsystems, timestamps that do not quite agree aligned by hand, what must have been present inferred from what was written down. Careful work by serious people, and still an argument about evidence rather than evidence.

That is because logs are written to help somebody debug, which leaves three gaps that matter afterwards. **They record what arrived**, so a reading expected and never received leaves no entry — frequently the most diagnostic fact in the interval. **They often do not record what was discarded, or why**, so six months later the rejection is invisible and the output looks like a clean decision on complete data. And **they can confuse when a thing happened with when it was heard about**: a reading taken at 14:02 and received at 14:05 is not the same fact as one taken at 14:05.

None of this is universal. Well-instrumented systems record rejections, carry both timestamps, and emit heartbeats that make silence visible. A fourth problem lives outside the log file: **an ordinary log can generally be edited afterwards by anyone with the access to do so**. For a debugging tool that hardly matters. For evidence it is the whole question.

What this capability does

Every input that reaches the recorded ingestion boundary gets exactly one recorded outcome. Used, held, refused, or too late to count. Nothing crossing that boundary is silently dropped, so the account of **received** inputs is complete with respect to that boundary.

Nothing received from an expected source is recorded separately, as its own kind of entry: received inputs are accounted for one by one, while a silent source is recorded as *nothing received from this source in this interval*. Two different facts, kept apart.

"Nothing received" is not "nothing sent." A source can transmit into a failed cable or a driver that discarded the data, and the record cannot tell that from a source that transmitted nothing; separating them needs evidence from the sending side — a transmit log, an acknowledgement, a link counter — which this does not supply. The record establishes that the system had nothing from that source, and when.

When a thing happened is kept separate from when it was heard. A reading arriving late is handled against the moment it belongs to, not the moment it turned up.

The record replays deterministically, under stated conditions. Given the same recorded inputs, the same recorded configuration and the same version of the decision logic, re-running an interval reproduces the same record identically rather than approximately. Those conditions are what make it a check: a replay reproduces **what the system did under the configuration it was running**, and is not a counterfactual — it does not answer what the system *would* have done under a different configuration or a later software version.

The record is hash-chained, so alteration is detectable. A later edit breaks the chain, found by recomputing it against a separately held anchor. Editing is **detectable, not impossible**, and the strength of the detection rests on where the anchor lives: one held by the party that holds the record can in principle be recomputed along with it, while one written at the time to a counterparty or an append-only external service cannot. That is a deployment decision made before an incident, and what an investigator should ask about first.

An illustrative scenario

This is not a customer story. No deployment described here has taken place.

An automated action is taken at 14:07. Nine weeks later it is disputed, and the operator has to explain it.

Without this: three subsystems' logs, timestamps that agree to within a second or two, and a gap where a fourth source should have been contributing, with no way to tell whether it was broken, blocked, or not fitted that day. Two sets of engineers produce two careful accounts, and the dispute is now about the reconstructions.

With it: the interval is re-run under its recorded configuration and produces the same record — which sources contributed, which was expected with nothing received from it, which arrived late, and what was done with each.

Note what that does not promise. It does not prove the decision was correct. It establishes what the decision was made on, and — to the extent the anchoring supports — that the account has not been altered since. A smaller claim, and a more useful one, because it can be checked. Establishing those facts is much of the work in a review, and the same record is a defence when the automation behaved correctly. That is design intent, not a measured outcome: nothing here has been assessed by an insurer, a regulator or a class society.

What this is not

- **Not proof that a decision was right.** It establishes what the decision was made on. Whether that was correct is a separate judgement, which this is meant to make possible rather than to make for you.
- **Not a complete account of the world, and not a counterfactual simulator.** Completeness is bounded by the recorded ingestion boundary, and replay reproduces the recorded run under its recorded configuration.
- **Not tamper-proof.** Alteration is detectable by recomputation against a separately held anchor, and correspondingly weaker where that anchor sits under the same control as the record.
- **Not a compliance product, and not legal advice.** It implements no regulatory scheme, no regulator has assessed it, and nothing here is a statement about the admissibility or evidential weight of any record in any jurisdiction.
- **Not yet proven in the field.** Results to date are from simulation. No field trial, no installed base.
- **Not a granted patent.** Four US provisional applications are on file. None has been examined and none is granted.

A first conversation. Bring a nonconfidential example — a review where establishing what the system knew turned out to be the hard part. We will work through whether this architecture would have changed it and what a joint evaluation would need to establish. Recorded data, if it helps, comes later and under terms you set. A first call is a technical discussion, not a pitch, and we will say plainly when the answer is no.

info@rvtechconsulting.com · RV Technology Consulting, Las Vegas, Nevada

Covered by US provisional patent application 64/159,476, filed 21 September 2026. Four US provisional patent applications filed 2026. Patent pending; none examined or granted. The scenario above is illustrative and describes no actual deployment. This document describes capability only and is not a technical specification.