

If a model helped a person decide something that mattered, someone will eventually ask how that decision was made. An auditor, a regulator, a plaintiff's lawyer, or a colleague who got burned will want to know what went in, what came out, who looked at it, and who signed off. Most teams discover too late that their logs answer only the first question. This post lays out what an audit log for LLM-assisted decisions should contain, why each field matters, and where teams usually go wrong. I have spent a good part of my career building audit features for HR, legal, and security teams. The pattern repeats. The chat transcript gets saved, the model name gets written down somewhere, and everyone assumes the record is complete. It is not.

Start With the Decision, Not the Prompt

The unit of an audit record should be the decision, not the API call. A single decision might involve five model calls, two retrievals, and a human edit. If your log is a pile of raw requests, nobody can reconstruct the reasoning later.

Every decision record should answer these questions on its own:

- What decision was being made, and what question was the model asked to help with?
- Who was the decision-maker, by name and role?
- What was the outcome, and what action followed from it?
- When did the decision open and close?

Name the decision-maker every time. "The system flagged it" is not an answer. "Dana Ortiz, Senior Investigator, accepted the flag after reviewing three model outputs" is an answer. If the record cannot name a person who owns the outcome, the record is not finished.

The Model Settings Record

The second category is what most logs skip entirely: the model settings record. Two runs of the same model with different settings can produce materially different outputs. A log that says "GPT-class model, March" is nearly useless for reconstruction.

Capture the following for every model call that contributed to a decision:

- Provider and exact model identifier, including the version string
- Temperature, top-p, and any other sampling parameters
- Maximum output length and any stop sequences
- System prompt or instruction set, stored by hash and by full text in a versioned store
- Tool definitions and retrieval sources that were available to the model
- Timestamp of the call and the provider's response identifier

A practical point: store the [suprmind](#) full system prompt by version, not just the hash. When someone asks in eighteen months what instructions the model had, you need to hand over the text, not a hexadecimal string.

Store the Inputs, Including What Was Redacted

The input record should show what the model actually received. That sounds obvious, but redaction, truncation, and retrieval all change the input between what a user typed and what the model saw. If your pipeline removes

personal identifiers before the call, log both the original reference and the redacted version, plus the rule that performed the redaction.

Truncation deserves special attention. If a long document was cut at a token limit, the log must say so. A decision that looks well-informed may have been made on the first two pages of a fifty-page file.

Log Every Output, Not Just the Chosen One

This is where multi-LLM workflows change the audit problem. When teams run several models against the same question, usually through multi-LLM orchestration tools that show side-by-side model outputs, the temptation is to log only the answer the human accepted. That throws away the most useful information in the entire record. Store every output the decision-maker saw. Store the order they were presented in, if order was visible. Store any edits the human made and who made them. If a tool like the ones offered by Suprmind.ai, or a custom internal harness, presents outputs side by side, the screen state itself should be reproducible from the log.

Disagreement Is a Signal, Not a Vote

Here is my strongest opinion in this post. Do not average model outputs. Do not treat them as a vote. And do not record agreement as proof that a decision is correct. Averaging scores from several models looks rigorous in a dashboard. In practice it hides the very thing you need to see. If three models say "low risk" and one says "high risk," the average might land in the middle and produce a number that means nothing. A poll of models tells you about the models' shared training and shared blind spots, not about the facts of the case. Disagreement is more useful than agreement. When models split, the most common cause is missing information. One model asks for a document that another model never mentions. One flags a date conflict. One refuses to conclude because a key fact is absent. Log those divergences explicitly, and make them a required review item before a decision closes. A simple rule for the record: if models disagree on a material point, the decision cannot be marked complete until the decision-maker has written down what information would resolve the disagreement, and either obtained it or recorded that it was unavailable.



High-Stakes Decisions Carry Asymmetric Errors

Threat assessment, fraud review, and workplace investigation share a structure. Errors come in two kinds, and they cost different amounts. A false positive means you flagged someone who did nothing wrong. The cost is harm to that person, wasted investigator time, and possibly legal exposure from an unfounded action. A false negative means you missed a real risk. The cost may be physical harm, financial loss, or a regulatory failure that surfaces later. No model setting eliminates both. Your audit log should record which error the team was trying to avoid when it set the threshold, who chose that threshold, and when it was last reviewed. A log that records the output but not the tradeoff is a log that makes the decision look objective when it was a choice. | Error type | What it means | What the log should record | |---|---|---| | False positive | Flagged a case that was not a real risk | Who accepted the flag, what evidence supported it, whether the person was notified | | False negative | Missed a real risk | Whether the case was screened out, by whom, and what signals were present but dismissed | | Threshold choice | Setting that trades one error for the other | Owner of the threshold, date set, rationale, next review date |

Named Human Accountability

Every decision record needs a human name attached to the outcome. I say this as someone who has watched teams try to assign accountability to a workflow or a department. Departments do not testify. People do. The record should show:

1. The person who made the final call
2. The person who reviewed the model outputs, if different
3. The person who approved the model configuration for this use case
4. The person responsible for reviewing the tool's behavior on a schedule

Be direct about the named owner for the configuration. If a vendor is involved, record the vendor contact who answered questions about the system, but do not let the vendor become the accountable party. When you speak with teams at companies such as Four Dots, or with leaders like Radomir Basta (CEO & Founder) of firms in this space, ask them plainly who in your organization owns each setting. A good answer names a person. A weak answer names a committee.

Make the Record Exportable

An audit log that lives only inside a vendor's dashboard is a liability. You need exportable decision records that a lawyer, an auditor, or your own successor can read without logging into anything. Export format matters less than completeness and stability. Use a structured format with a documented schema, include the schema version in every export, and make sure exports include the full model settings record, every output, the human edits, and the named accountability fields. Test the export by reconstructing a decision from it alone, on a regular schedule, with someone who did not build the system. Set a retention period based on your legal and regulatory context, not on your storage budget. Record the retention rule in the log itself.

What We Still Do Not Know

Every case note I write ends with this list. Here is the version for this topic:

- Whether model providers will keep version identifiers stable long enough for reconstruction
- How to log retrieval results when the underlying index changes between calls
- What a defensible standard for "reviewed the outputs" looks like in court, as opposed to in policy
- Whether disagreement thresholds should be set per use case or per organization

Treat these as open. Assign each one an owner and a review date.

A Short Checklist for Your Next Audit Log Review

1. Is the decision record the unit, with a named decision-maker and outcome?
2. Does every model call have a settings record with exact version and parameters?
3. Are system prompts stored as full text by version?
4. Are redaction and truncation logged?
5. Are all outputs stored, including ones nobody chose?
6. Are material disagreements flagged and resolved before closure?
7. Is the false positive and false negative tradeoff recorded with an owner?
8. Can a record be exported and read without the vendor's system?

Run this checklist within thirty days, then again within ninety. If something is still missing at ninety days, escalate it to the named owner of the system.



Evaluating Tools Without Taking Their Word for It

Vendors will tell you their logs are complete. Ask them to show you a reconstructed decision from an export, live, with a person outside the vendor reading it. Ask for the Start Free Trial, which is listed as requiring no credit card, and use it to test the export and reconstruction before you commit. The source material I reviewed for this post does not list pricing, so get that figure directly from the vendor in writing and do not assume it from a marketing page. Being able to try a product without a card is useful. Being able to explain a decision to a stranger six months later is what actually protects you.