Seven questions your agent should answer before an auditor asks

Seven questions, each answerable on your own system this week without asking a vendor anything. They are engineering questions: each one asks whether a property was built deliberately, because none of them arrive with a framework and most cannot be added retroactively.

The obligation families are named so you can route each finding to the right person. We build systems; your counsel interprets the regulation.

1 RECORD-KEEPING

Take one decision your system made last month. Without opening a terminal: what exactly did the model have in front of it at that moment?

Good	The messages sent to the model are in the record, verbatim, at that step.
Weak	You hold the state and could rebuild the prompt — if the template has not changed.
Failing	“We’d need the developer who built it.”

Every other question depends on this one. If the input cannot be reconstructed, neither the output nor anything downstream can be evaluated.

☐ Good ☐ Weak ☐ Fail

2 HUMAN OVERSIGHT

Who approved the last gated action your system took, and what evidence exists that they were entitled to approve it and saw enough to decide?

Good	An identity in the decision record, the summary they were shown, elapsed time, and a denial that terminates the action.
Weak	An identity correlated from a web access log with its own retention policy.
Failing	A boolean. A resume value records that something sent true.

Follow-up worth running: what is your approval rate? Near 100% means either the gate is mis-scoped or nobody is reading. Oversight degrades through volume, not negligence — and only shows up in data you chose to record.

☐ Good ☐ Weak ☐ Fail

3 CONTROL EFFECTIVENESS

Delete the rule that requires approval for your riskiest action. Run the tests. How many went red?

Good	A red test named after the guarantee — one a reviewer can read without reading your codebase.
Weak	An integration test went red. Asserted incidentally; it will be rewritten when the flow changes.
Failing	Nothing red. That is a habit, not a control.

A habit is behaviour that is currently correct. A control is behaviour that cannot silently stop being correct. Under normal engineering pressure the two are indistinguishable; they diverge at the next refactor and under review.

☐ Good ☐ Weak ☐ Fail

4 HUMAN OVERSIGHT

Find the last time your agent was uncertain. What did it do?

Good	Two independent channels: the agent can escalate, and detectors fire on observable symptoms — repeated tool failures, empty retrievals, a customer repeating themselves.
Weak	The agent can escalate if it decides to.
Failing	You cannot find an instance, because uncertainty is not recorded as anything.

One channel is not enough. An agent that is confused is frequently confused in a way it cannot perceive, so asking a stuck model whether it is stuck is unreliable exactly when it matters.

☐ Good ☐ Weak ☐ Fail

5 THIRD-PARTY RISK, EXIT READINESS

Your AI vendor terminates tomorrow. What file arrives, and can you open it with nothing but a standard library?

Good	Append-only, one row per event, human-legible. Answers “what was decided on 14 March and why” with no vendor code present.
Weak	A database dump. Readable, but records where the system was rather than what it decided.
Failing	Checkpoints in a framework’s serialisation format — a copy of your dependency.

Second-order question people skip: who operated it? A perfect export of a system you cannot run leaves you with evidence and no capability.

☐ Good ☐ Weak ☐ Fail

6 ACCURACY AND ACCOUNTABILITY

How do you know your agent answered the question that was asked, rather than an adjacent one?

Good	A check against the original request, run by something that did not do the work, with “wrong question” as a verdict distinct from “incomplete”.
Weak	Schema validation. Confirms shape, not subject.
Failing	Nothing checks — which is the default.

Every standard guard watches process: budget, retries, loops, permissions, cancellation. None asks whether the right question was answered. In document work the failure is rarely invention; it is faithful quotation from the superseded version.

☐ Good ☐ Weak ☐ Fail

7 TRANSPARENCY AND VENDOR REVIEW

Can your risk reviewer read the thing that makes decisions, or only the thing that calls it?

Good	Decision logic is a small amount of code, readable in an afternoon, with few dependencies. A dependency is something your reviewer must also read.
Weak	A framework they must understand first — its execution model, its state semantics — before they can understand your audit trail.
Failing	Logic spread across prompts, graph topology and library internals. The honest answer is “you would have to trust us”.

Last because every other answer is delivered through it. An excellent control nobody can verify is an excellent control you cannot evidence.

☐ Good ☐ Weak ☐ Fail

Reading your score

0–2 failing	You can evidence most of what a reviewer will ask for. Work on the weak answers before the failing ones: a weak answer usually means a control exists but is asserted incidentally, which is cheaper to fix than absence.
3–4 failing	Typical for a system built for delivery speed, which is the correct goal for most of its life. The question is whether the failures are the retroactive kind — questions 1, 2 and 5 record information that cannot be recovered later.
5 or more	The system was not built to be reviewed, and reviewing it will be a project rather than an exercise. That is worth knowing before an audit date makes it urgent.

Where we fail this ourselves

We ran the seven against our own work while writing them. Three failures, named, because a checklist that only points outward is marketing.

Retention and redaction

Our record keeps everything verbatim, for ever. That is the right default for evidence and the wrong one for personal data. Field-level redaction at write time, with replay still intact afterwards, is in progress. Today it is a decision at ingestion, which is weaker. This is the gap we would raise first if we were reviewing us.

Authority is asserted, not verified

We record who approved. We do not verify they were entitled to. Binding approvals to a role model in your IAM is the right design and remains an integration you would write. We also cannot express a second-signature requirement, which some high-value actions want.

Nothing alerts on a degrading control

We record enough to detect an approval queue being rubber-stamped. Nothing watches for it. Detecting a failing control and telling somebody are different features, and we have built the first.

"We improve continuously" is undemonstrable, which is why it is worthless. The version that means something is a changelog. From the weeks spent writing this, caught by tests rather than by review:

- a filter deleting valid records silently — precision 100%, recall 54%, and nothing in the output looked wrong
- two tautological assertions in our own test code, of a form that cannot fail
- a suite silently skipping every async test, because a plugin was undeclared — green throughout
- an approval gate whose explicit rule proved redundant against an undocumented default

What to do with this

Run the seven against your own system, then against whoever builds your AI, and compare the deltas. The exercise is worth more than any vendor's answer, including ours. If it turns up gaps you would rather not close yourself, we do this for authorised firms under MiCA, DORA and the EU AI Act — 30 minutes, one obligation, an honest answer about what production would take.