

Twelve questions to ask before August 2, 2026.

Hand this to your compliance vendor. Klaritee passes all twelve. The questions your vendor cannot answer are the ones a regulator will press.

THE QUESTION	SMALL / MID <\$585M	LARGE \$5.8–11.7B	ENTERPRISE >\$23B	KLARITEE	YOUR VENDOR?
PART ONE · THE SIX REGULATORY INVARIANTS					
Q1 Pure deterministic rules, with no LLM in the evaluation path?	\$12–18M	\$175–350M	\$700M–1.6B	✓	☐
Q2 Character-level sourceSpans on every finding (byte coordinates)?	\$12–18M	\$175–290M	\$525M–1.05B	✓	☐
Q3 Entire evidence chain exportable as one unified raw JSON document?	\$9–18M	\$140–235M	\$470–935M	✓	☐
Q4 Cryptographic SHA-256 receipt binding input, engine version, and output?	\$12–18M	\$175–350M	\$700M–1.4B	✓	☐
Q5 Audit trail anchored to a public, immutable, vendor-independent ledger?	\$8–14M	\$115–235M	\$350–820M	✓	☐
Q6 Byte-for-byte reproducibility, same hash two years later?	\$18M	\$350M+	\$1.2B+	✓	☐
PART TWO · THE SIX <code>Record<string, any></code> QUESTIONS					
Q7 <code>Record<string, any></code> absent from the production codebase?	\$9–18M	\$175–350M	\$585M–1.17B	✓	☐
Q8 Scoring engine free of the unguarded type (not just the display layer)?	\$18–41M	\$350–820M	\$1.6–4.1B+	✓	☐
Q9 No insurance-coverage gap from a faulty type-driven verdict?	Full fine + defense costs · all tiers			✓	☐
Q10 Terms of service transparent about this pattern, with no buried liability shift?	\$12–23M	\$235–585M	\$820M–2.3B+	✓	☐
Q11 Line-by-line audit report available, with file paths and line numbers?	\$9–18M	\$175–290M	\$470–935M	✓	☐
Q12 Ready to answer a regulator on type architecture? (Articles 9, 12, 17.)	\$12–23M	\$235–470M	\$600M–1.4B	✓	☐

Expect a standardized playbook.

When pressed on type safety, many vendors deploy one of three rehearsed defenses to mask a structural vulnerability. **Here is the first, and what it means.**

1

The "Defense in Depth" pivot

WAF & INGRESS MISDIRECTION

► THE SPIN

"Our internal evaluation logic uses dynamic index signatures for high-throughput processing, but it is entirely protected by external ingress validation. We use strict JSON Schema validators and a Web Application Firewall at the API boundary, so malformed data never reaches the scoring engine."

▼ THE REALITY

They are passing off network security as application-layer determinism. **If an internal model extraction layer drifts and outputs a structural anomaly, a `Record<string, any>` scoring engine ingests it blindly, without objecting.** The firewall never sees it, because the anomaly originates inside the system, past the gate.

A guard at the front gate does not fix a structural crack in the vault door.

The runtime guard illusion.

The second defense sounds the most technical, which is exactly why it works on non-engineers. **Here is what it conceals.**

2 The "Runtime Type Guard" illusion

PARTIAL VALIDATION DRESSED AS COMPLETE

► THE SPIN

"Our TypeScript definitions use flexible typing for developer velocity and schema evolution, but we enforce strict runtime validation libraries, like Zod or Yup, and functional type guards at execution time to ensure data integrity."

▼ THE REALITY

This is being half-pregnant. They may run runtime checks on broad fields, like confirming a user ID is a number, but they rarely enforce exhaustive, closed-type validation across every internal scoring parameter. **If the runtime guard were genuinely complete, the TypeScript definition would not need to be an open any. It would be a strictly mapped, concrete typed struct.**

A static type wildcard is how they hide a missing internal lineage.

The certification shield.

The third defense answers a code-integrity question with an administrative certificate. **They are not the same thing.**

3

The "We are SOC 2 / ISO 27001 / ISO 42001 certified" shield

ADMINISTRATIVE COVER FOR A CODE QUESTION

► THE SPIN

"Our platform undergoes rigorous third-party penetration testing and holds gold-standard certifications, including SOC 2, ISO 27001, and ISO/IEC 42001:2023 for AI management systems. Our compliance posture is fully validated."

▼ THE REALITY

SOC 2 and ISO 27001 check whether you encrypt data at rest, enforce a password policy, and log developer access. ISO/IEC 42001 audits the management system around your AI: governance, risk processes, documentation. **None of the three compile the codebase to check whether an engineer left a gaping vulnerability at an explicit mathematical scoring boundary.** An AI governance certificate is not a proof of code integrity.

An administrative certificate cannot answer a question about the scoring engine's type.

WHEN YOU HAVE HEARD ALL THREE DEFENSES

If the guardrails are as bulletproof as they say, why keep the escape hatch?

Every defense rests on the same claim: the open type is safe because something else protects it. **So ask them to remove it.** There is only one honest answer to this question.

ASK THIS. WORD FOR WORD.

"Can you push a commit today that replaces `Record<string, any>` with a closed string-union type and a concrete typed struct, and **confirm in writing, contractually, a zero-error build log before the August 2, 2026 deadline?**"

THEN CLOSE THE LOOP

- Get the answer **in writing**. A verbal assurance is not a record.
- Document it and **file it away**, ready at a moment's notice by the deadline.
- Review your **terms and conditions** for where liability was shifted to you.
- Start the conversation **now**, not in the audit.

If the escape hatch was never load-bearing, removing it costs nothing. If they refuse, you have your answer.