

ONE QUESTION YOUR COMPLIANCE VENDOR CANNOT ANSWER

Is `Record<string, any>` running in your scoring engine?

Security researchers have documented that this single TypeScript pattern disables type checking at critical data boundaries. Most compliance vendors have never looked. One has. The results are below.

WHY THIS MATTERS NOW

High-risk AI system obligations under EU AI Act Articles 9–17 become enforceable August 2, 2026. Penalties reach EUR 35 million or 7% of global turnover. The type system your vendor uses is not a style choice. It is an audit risk.

WHAT IT IS

The type that tells the compiler to stop checking.

One line of TypeScript. Entire data boundary left unguarded.

```
Record<string, any>
```

- **string** means any key is accepted. Any property name. From any source.
- **any** means any value is accepted. Any type. Any shape. Including injected, malformed, or fabricated data.
- When this sits at a **scoring boundary**, the engine processes whatever arrives. The type system does not object. The score is produced. **You never know.**
- Security researchers at OWASP and Google Project Zero have documented this pattern as a **structural vulnerability at data boundaries**. Not a style preference.

SIX QUESTIONS. HAND THIS TO YOUR VENDOR.

Ask before you sign. Ask before you submit.

Print this. Screenshot this. Email this to your compliance team today.



01

Is `Record<string, any>` present anywhere in your production codebase?

Not staging. Not test. Production. The code that produces the compliance output your regulators see.



02

Is that type in the engine that calculates the score, or only in the interface that displays it?

These are different risks. A display-layer type issue is cosmetic. A scoring-engine type issue means the number itself is unguarded.



03

If this pattern produced a faulty compliance verdict and we were fined, would our cyber insurance cover it?

Ask your insurer the same question. Get the answer in writing.



04

Does your terms of service explicitly disclose this type pattern or limit liability for outputs it may affect?

If it is buried in the indemnification clause, you accepted the risk without knowing.



05

Can you provide a line-by-line audit report confirming where this type does and does not appear?

A claim is not an audit. Ask for the file paths, the line numbers, and the code context.



06

Will an EU AI Act regulator ask us about the type architecture of our compliance system?

Article 9 technical documentation requirements. Article 12 logging. Article 17 quality management. The answer is yes.

TYPE ARCHITECTURE CHECK

A regulator pulls your vendor's code. What do they find?

The type that processes your compliance data is either guarded or it isn't.

THE REGULATOR ASKS

"Show me the TypeScript type that sits at your scoring boundary. Is it concrete and closed, or does it accept anything?"

VENDOR A

A workflow platform

The scoring boundary uses **Record<string, any>**. The **type system looks away**. Any property, any value type, any shape. All accepted without check. The score is produced from whatever arrives.

THEN THE REGULATOR PUSHES

"What validated this data before it reached your scorer?"

RESULT

No answer. The type itself is the gap. Audit stalls.

WHAT IS MISSING

A closed type at the scoring boundary. One that proves every field was validated, named, and expected before the engine ran.

VENDOR RUNNING KLARITEE

A deterministic compiler

A 15-chunk, line-by-line audit across 221 TypeScript files. **Zero instances** of Record<string, any>. Every scoring boundary uses closed enums and concrete typed structs. The compiler enforces it.

THEN THE REGULATOR PUSHES

"Show me the audit report."

RESULT

Full file-path-level report produced. Zero instances confirmed. Audit closes.

WHAT IS DELIVERED

A verifiable type-safety audit. Every boundary named. Every type closed. Every result traceable to a concrete, validated input.

WHAT THIS PATTERN ACTUALLY COSTS

Four consequences your vendor will not volunteer.

These are not hypothetical. They are structural outcomes of the pattern.

01 The score is not provably derived from your document.

When the type system accepts any shape, there is no formal proof the scoring input came from your text and nothing else. **A regulator asking for traceability gets a silence.**

02 Your insurance may not cover a fine caused by a vendor's type architecture.

Cyber insurers are now reviewing AI system liability clauses. **If the compliance output was structurally unguarded and a fine results**, the coverage question is not hypothetical. It is an open one.

03 The terms of service may have already transferred this risk to you.

Vendors who know this pattern exists often disclaim liability for output accuracy in clause 14. **You accepted it at signature.** Most compliance buyers do not read that far.

04 Article 9 technical documentation cannot be satisfied by a dashboard screenshot.

EU AI Act requires documented evidence of technical robustness. A type-unguarded scoring engine produces output with no formally verifiable input chain. **That is not Article 9 compliant.**

IF YOUR VENDOR SAYS THIS

“It's no big deal. Everyone uses that pattern.”

"That's standard TypeScript. It doesn't affect accuracy. You're overthinking it."

— EVERY VENDOR WHO HAS NEVER RUN THE AUDIT

WHAT THAT ANSWER MEANS

- They **have not looked**. "Everyone uses it" is not an audit. Ask for the file paths and line numbers.
- "Doesn't affect accuracy" is an assertion with no proof path. Ask them to show the type that validates the input before the scorer runs.
- A regulator does not accept "**standard practice**" as a technical documentation artifact under Article 9. Your vendor's confidence is not your compliance.
- The companies that will be cited first under August 2026 enforcement are the ones who asked the vendor and **accepted a talking point instead of evidence**.

WE RAN THE AUDIT. HERE ARE THE NUMBERS.

221 files. 15 chunks. Zero instances.

Not a claim. A line-by-line report with file paths and line numbers.

CLAUDE CODE AUDIT / KLARITEE PRODUCTION CODEBASE

221

TypeScript files reviewed

0

Record<string, any> instances
Production and test. Every variant.

15

Sequential audit chunks

Every scoring boundary uses **closed string-union types and concrete typed structs**. The composite score is a deterministic weighted sum.
Unknown keys resolve to undefined and are guarded. The type system is a guard, not a gap.

API entry boundary (request validation)	CLEAN
Model extraction layer (text → structured data)	CLEAN
62 detectors (core decision path)	CLEAN
12 integrity indicators (enhanced decision path)	CLEAN
Scoring and tolerance layers	CLEAN
Integrity engine (all 11 sub-scores)	CLEAN
LLM boundary (only external data input)	FIELD-VALIDATED + WHITELISTED

THE QUESTION TO ASK EVERY VENDOR

Which vendor can prove their **type system** is a guard?

The rest of the market is producing compliance outputs. Klaritee is producing compliance outputs with a closed, audited, formally typed scoring engine. And the report to prove it.

The companies that survive the August 2026 enforcement wave will be the ones who **asked for evidence**, not assurances.

IF YOU BOUGHT ONE OF THE VENDORS ON THIS LIST

Run the six due diligence questions from slide 3 against the platform you already own. The ones that cannot answer are the ones a regulator will eventually press. **Then we should talk.**