

The clock the entire compliance industry is racing.

67

days left
until August 2, 2026

Ten leading vendors. Six requirements that actually matter. One question they cannot answer.

High-risk AI system obligations under Articles 9 through 17 and Article 26 become enforceable on August 2, 2026. Penalties reach up to EUR 35 million or 7 percent of global turnover.

Ten vendors. Six requirements. Six failures.

When you strip away the marketing, every leading EU AI Act compliance platform fails the same six tests.

INVARIANCE CHECK	STATUS	WHY THEY FAIL
1. Pure deterministic rules No LLM in the evaluation path	FAILED 8 of 10	Optro, Vanta, and Credo AI use generative LLMs to interpret documents and decide if evidence complies.
2. Character-level sourceSpans Byte-coordinate anchors to source	FAILED 10 of 10	None parse evidence down to exact character offsets. Page-level attachments fail Article 12.
3. Evidence in raw JSON Flattened, exportable, unified	FAILED 10 of 10	Compliance data lives inside a dashboard, not a JSON document binding clauses to specific string assets.
4. Cryptographic receipt SHA-256 binding input and output	FAILED 9 of 10	Audit data sits in cloud tables with no permanent hash binding input, engine version, and output.
5. Public immutable ledger Blockchain-anchored verification	FAILED 10 of 10	Centralized private SaaS databases. If an admin can edit the row, the audit trail is not immutable.
6. Byte-for-byte reproducibility Same hash twelve months later	FAILED 10 of 10	Not one vendor commits to public reproducibility. Cloud updates and LLM drift break the hash.

A REGULATOR WALKS IN

One question. Two systems. Two very different answers.

It is **August 3, 2027**. A national market surveillance authority opens an Article 12 audit on your high-risk AI system.

They ask one thing. "Re-run the compliance evaluation you submitted twelve months ago. Show us the same result."

Which vendor do you want sitting next to you when the regulator says "go"?

Run the same document twice. Do you get the same verdict?

Pure deterministic rules. No LLM in the evaluation path.

THE REGULATOR ASKS

"Re-evaluate this filing. Today. Same engine. Same document. Show me the result."

VENDOR A

A workflow platform

The platform routes the document through a generative LLM that **interprets** the contents and **guesses** whether evidence complies. The model has been updated three times since the last run. The temperature parameter is non-zero. The result shifts.

THEN THE REGULATOR PUSHES

"Why did the verdict change?"

RESULT

Different verdict than twelve months ago. Audit fails.

WHAT IS MISSING

A reproducible answer to the question "is this the same engine that ran last year?"

VENDOR RUNNING KLARITEE

A deterministic compiler

The document is parsed by a **deterministic rules engine**. No LLM in the evaluation path. The same input plus the same engine version always produces the same output. Bit-identical.

THEN THE REGULATOR PUSHES

"Prove it didn't change."

RESULT

Identical verdict. Reproducible on the spot.

WHAT IS DELIVERED

A pinned engine version with a hash that resolves to the exact rules, the exact code, and the exact verdict.

Where exactly in the document does this finding come from?

Character-level sourceSpans. Byte coordinates that pin every finding to exact ranges.

THE REGULATOR ASKS

"Show me the exact sentence in the source that triggered this flag. By character offset."

VENDOR A

A workflow platform

The platform points to **page 4, paragraph 2** of a PDF attachment. The actual finding was generated by an LLM that paraphrased the source. The regulator cannot map the finding back to a verifiable span of text.

THEN THE REGULATOR PUSHES

"Which exact sentence? Show me the offset."

RESULT

Cannot prove the finding originated from the document.

WHAT IS MISSING

A coordinate. There is no way to point at the exact bytes that triggered the flag.

VENDOR RUNNING KLARITEE

A deterministic compiler

Every finding carries **byte-coordinate sourceSpans**: start offset, end offset, hash of the underlying string. The regulator opens the source, jumps to characters 18,442 through 18,506, and sees the exact sentence.

THEN THE REGULATOR PUSHES

"Verify the hash of that span."

RESULT

Finding bound to source. Reconstruction satisfied.

WHAT IS DELIVERED

Start offset 18,442. End offset 18,506. SHA-256 of the underlying string. Verifiable in a regulator's terminal.

Can you export the entire evidence chain as one machine-readable document?

Evidence anchored back to raw, flattened, unified JSON.

THE REGULATOR ASKS

"Send us the full evidence chain. Machine-readable. Independently parseable."

VENDOR A

A workflow platform

The platform exports a **PDF report** generated from a web dashboard. The underlying data lives across seventeen database tables the customer cannot access. Reconstructing the chain requires a vendor support ticket.

THEN THE REGULATOR PUSHES

"We need the raw data, not a printout."

RESULT

Evidence is shown, not produced. Chain breaks.

WHAT IS MISSING

A document. The regulator cannot leave with the evidence. They can only leave with screenshots.

VENDOR RUNNING KLARITEE

A deterministic compiler

A **single unified JSON** binds every regulatory clause to every detector occurrence to every byte coordinate to every score. The regulator's team parses it in their own environment in seconds.

THEN THE REGULATOR PUSHES

"Parse this in our environment."

RESULT

Full chain produced. Independently verifiable.

WHAT IS DELIVERED

One JSON file. Every clause, every occurrence, every offset, every score. The regulator parses it themselves.

Prove the result on the screen matches the result you submitted last year.

Cryptographic verification receipt. SHA-256 binding input, engine version, and output.

THE REGULATOR ASKS

"How do I know this dashboard hasn't been edited since submission?"

VENDOR A

A workflow platform

The audit log lives in a cloud database table. An enterprise admin has edit rights. The vendor's terms of service permit retroactive corrections. There is no permanent cryptographic binding between the original input and the displayed output.

THEN THE REGULATOR PUSHES

"Who could have edited this row?"

RESULT

Audit trail is editable. Trust collapses.

WHAT IS MISSING

Cryptographic proof. The audit log is a database row, not a signed artifact.

VENDOR RUNNING KLARITEE

A deterministic compiler

Every evaluation produces a **SHA-256 receipt** binding the input, the engine version, the timestamp, the run ID, and the full output into a single permanent artifact. The hash is in the regulator's hands.

THEN THE REGULATOR PUSHES

"Verify the hash against our copy."

RESULT

Receipt re-verifies. Match is mathematical.

WHAT IS DELIVERED

A SHA-256 receipt binding input, engine version, timestamp, run ID, and output. Held by the regulator. Re-verifiable in seconds.

INVARIANCE 5. LEDGER

If the vendor disappeared tomorrow, could the audit history survive?

Blockchain-anchored public ledger. Externally verifiable. Vendor-independent.

THE REGULATOR ASKS

"Verify this submission against a record the vendor cannot alter."

VENDOR A

A workflow platform

The submission lives entirely inside the vendor's private SaaS database. If the vendor changes pricing, gets acquired, suffers an outage, or quietly edits the row, the audit trail is gone. The customer has no independent record.

THEN THE REGULATOR PUSHES

"Show me a copy the vendor cannot touch."

RESULT

Audit history is hostage. No external proof.

WHAT IS MISSING

Independence. If the vendor goes away, the audit trail goes with them.

VENDOR RUNNING KLARITEE

A deterministic compiler

Every receipt is **anchored to a public ledger**. The vendor cannot rewrite history. The customer cannot rewrite history. An auditor verifies the hash against the chain in their own browser.

THEN THE REGULATOR PUSHES

"Open the ledger. Verify it now."

RESULT

Audit history is permanent. Vendor-independent.

WHAT IS DELIVERED

A public ledger anchor that survives the vendor, the customer, and the auditor. Verified in the regulator's browser.

It is two years from now. Re-run the check. Does it match?

Byte-for-byte reproducibility. Same hash. Years later.

THE REGULATOR ASKS

"Reproduce the August 2027 evaluation. Byte for byte. We'll verify."

VENDOR A

A workflow platform

The vendor's environment has been updated 47 times. The LLM prompts have been quietly rewritten. The API endpoints have changed twice. The model weights have shifted. The background data hash **cannot be reproduced**. The vendor offers a written explanation in lieu of evidence.

THEN THE REGULATOR PUSHES

"We don't want an explanation. We want the hash."

RESULT

Reproduction fails. Compliance claim collapses.

WHAT IS MISSING

Time invariance. Today's environment is not last year's environment, and there is no contractual commitment to bridge them.

VENDOR RUNNING KLARITEE

A deterministic compiler

The **engine version is pinned**. The rules are pinned. The output is deterministic. Two years later, the same input plus the same engine version produces the same hash. Bit-identical.

THEN THE REGULATOR PUSHES

"Re-run it. We'll check the hash."

RESULT

Reproduced exactly. Audit closes clean.

WHAT IS DELIVERED

A byte-for-byte reproducibility guarantee, contractually committed, technically enforced. Same hash today. Same hash in 2028.

THE QUESTION

Which vendor do you want sitting next to you when the regulator says "go"?

The one with a **SHA-256 receipt**, a **byte-anchored evidence chain**, and a **deterministic engine** that produces the same hash today, in twelve months, and in two years.

The rest of the market is helping companies write summaries about compliance.

Klaritee is compiling mathematically verifiable truth.

IF YOU BOUGHT ONE OF THE TEN

Run the six checks against the platform you already own. The ones that fail are the ones a regulator will eventually press. **Then we should talk.**