

FAQ

Provably Private AI + De-Identification

Frequently asked questions: a companion to the technical white paper *Provably Private AI as Infrastructure*

Hart Johnson | October 2026



A companion to the technical white paper "*Provably Private AI as Infrastructure*." Incident and regulatory references are drawn from third-party reporting and primary filings; they are cited and dated in the full paper.

[Read the white paper →](#)

The Basics

Q What is "zero-retention AI"?

AI processing where your input is used to generate a response and then not written to durable storage. No content database, no stored conversation history on the vendor's servers, no training on your inputs. The content exists on vendor infrastructure only for the few seconds the model is working with it in memory.

Q How is that different from "we don't train on your data"?

"We don't train on your data" is a policy about one *use* of data that is still being *stored*. Zero-retention is about not storing it at all. Stored data can still be breached, indexed by accident, subpoenaed, or exposed by an unrelated bug, regardless of whether anyone trained on it. If it isn't retained, none of those can happen.

Q Isn't "we won't retain it" also just a promise?

It's a promise only if retention is possible and policy is holding it back. The stronger version is architectural: no content datastore exists, the compute role is denied write access to storage, and logs are configured to exclude content. The goal is "we *can't*," verifiable by inspection, not "we won't," taken on faith.

Q Do you literally never see our data?

Under the current architecture, no, and we can show you why in the code. To produce a response, the model operates on your text in memory for a few seconds; that is processing, and it is unavoidable. Processing is not the same as our being able to see the content, and nowhere in the running system does your content come to rest for anyone to read:

- **It is never written to a store.** No code path writes document content to a database, object storage, the filesystem, or a cache. The data stores we use hold only metadata (who called which endpoint, usage counts, timestamps, billing identifiers).
- **It is never written to a log.** The logging layer serializes only an allowlisted set of operational fields, so even an accidental attempt to log your text is dropped rather than written.

- **The runtime cannot persist it even if code tried to.** The service's execution role has no permission to write to object storage or a log stream, so persisting content would take both a code change and an infrastructure permission change, not a setting we can flip.
- **A crash cannot carry it out.** Our error-reporting client is pinned to a version whose default does not capture local variables from stack frames, and we never enable that capture, so an exception traceback cannot ship a document variable off-box. The error context we do send is additionally stripped of content fields.

The one place your text leaves the service is the call to our model provider (Amazon Bedrock), which processes it in memory under a zero-data-retention posture: it does not store the input or output and does not train on it. So "we can't see it, by construction" is not a promise we are asking you to trust; it is a property your security team can verify from our source, our dependency pins, and our infrastructure policy.

Why This Matters

Q Why not just block AI tools, or trust the big vendors' enterprise tiers?

Blocking pushes employees to shadow IT and forfeits real productivity. And enterprise tiers still typically store content and depend on correct configuration. The last two years are full of examples where stored content became an exposure: a 2023 bug surfaced ChatGPT conversation titles and payment data; in 2025 and 2026 "share" features published ChatGPT, Grok, and Claude conversations (including medical records and credentials) to search engines and the Internet Archive; and in 2025 a court ordered OpenAI to retain even deleted logs. The common factor was retention.

Q What's the single strongest argument for this architecture?

A court can compel a vendor to retain and produce data it holds, even data a user asked to delete. No court can compel production of data that was never kept. Architecture survives legal orders, bugs, acquisitions, and misconfigurations that policy does not.

De-Identification

Q What does the de-identification feature actually do?

You give it a document; it returns a de-identified draft with identifiers removed or replaced by neutral placeholders (e.g., [NAME_1], [DATE_3]), plus a coverage report of what was detected, what was replaced, and what still needs human review. It targets the HIPAA Safe Harbor profile by default.

Q How does it find identifiers?

A hybrid pipeline: a deterministic pass (pattern-matching for structured identifiers like SSNs, dates, emails, phone numbers, account numbers), a model pass (for context-dependent identifiers like names in prose, places, and job titles that quietly identify someone), and a verification pass that re-scans the output and flags anything it couldn't confirm.

Q Does this make my document HIPAA compliant?

No. It produces a de-identified *draft* and a coverage report. HIPAA Safe Harbor also requires the covered entity's determination of "no actual knowledge" that the remaining information could identify someone, a human legal judgment the tool does not and cannot make. It makes that judgment faster and better-documented; it does not replace it.

Q So what's the human's job?

Review the side-by-side draft, resolve the items in the "needs review" list (where the system was uncertain), confirm no residual re-identification risk, and make the final determination. Free-text documents especially need this; identifiers hide in prose.

Q How are long documents handled?

They're split into overlapping, sentence-aware chunks, processed, and reassembled into one clean document. The chunking is designed so an identifier (or the label that disambiguates it) is never cut across a boundary: a small overlap window ensures each identifier appears whole in at least one chunk, and reassembly is done by mapping replacements back onto the original text by position, so overlaps collapse cleanly. Chunk boundaries are invisible in the result.

Q Can I use it without the Muse interface?

Yes. It's exposed as an API endpoint that returns the de-identified text and the structured coverage report, with the review disclaimer included in the response.

Q Does the coverage report itself leak the identifiers it found?

No. Excerpts of unconfirmed spans are partially masked (only the first and last few characters are shown), so the report tells a reviewer where to look without re-exposing the sensitive value.

Q What about identifiers in images, like a scanned ID or a photo?

Out of scope for the current feature, which operates on text. Don't assume embedded image content is addressed by text de-identification; handle that separately.

Verification & Assurance

Q How can our security team verify the privacy claims instead of trusting them?

That's the point of an architectural posture: it's inspectable. A reviewer can examine the running service for the absence of content storage and content logging, and can see directly that documents live in the browser's local storage rather than on the vendor's servers. Trust is replaced with inspection. We're glad to walk your security team through this during an evaluation.

Q Where do our documents actually live?

Client-side (in the browser), or in storage you already control if you choose to sync there. The vendor's servers don't hold a copy. That also means retention, access, and lifecycle of your stored documents are governed by *your* controls, so responsibility moves to where it belongs.

Q What about the underlying model provider?

Elixir processes AI requests through Amazon Bedrock, which does not store model inputs or outputs and does not use them to train models. The model operates on your text in memory to produce a response; the input is not retained.

Regulations

Q Is this only for HIPAA / healthcare?

No. HIPAA Safe Harbor is the worked example because it comes with a concrete, enumerable list of 18 identifiers. The same find → strip → verify → confirm pattern applies to GDPR (data minimization and storage limitation), CCPA/CPRA, GLBA and financial-sector rules, and the de-identification work the EU AI Act will keep making relevant as its obligations phase in (transparency since August 2026; high-risk obligations through December 2027 and August 2028). A zero-retention architecture is data minimization by construction.

Q Does this replace our compliance program or legal review?

No. It's infrastructure and tooling that make compliant workflows faster and lower-risk. Legal and compliance ownership stays with your organization. Anything you publish externally from this material should get your own compliance/legal review.

Boundaries & Honesty

Q What does this architecture NOT protect against?

The in-memory processing window (irreducible for any system that processes text); the security of your own storage if you sync documents there; endpoint and device security generally; and the correctness of a de-identification result, which is a human determination. It removes the "vendor was holding our data and something went wrong" class of risk, not every risk.

Q Are the attestations (SOC 2, HITRUST) in scope here?

Elixir holds a SOC 2 Type 2 and a HITRUST r2, but those are scoped to **Elixir Cloud** and are not claimed as covering the AI writing, image, voice, or API tools discussed here. Don't conflate them.

Q Why no accuracy or benchmark numbers?

Because we won't publish numbers we haven't measured. If and when we cite detection rates or throughput, they'll come from real evaluation, not estimates. The absence of a number is deliberate, not an oversight.



Hart Johnson

MSCSIA, CEH, CHFI · Executive Managing Director, DevSecOps

Hart Johnson leads DevSecOps at Elixir Technologies, integrating development, operations, and security across the platform's engineering organization. He holds an MS in Cyber Security and Information Assurance and is a Certified Ethical Hacker and Computer Hacking Forensic Investigator, with a background spanning DevOps, security and compliance, and engineering leadership in the SaaS industry.

Go deeper

The full white paper covers the reference architecture, threat model, and a worked de-identification example.

[Read the full white paper →](#)

elixir.com/provably-private-ai-white-paper

elixir.com
Ojai, CA

elixir