

Data Sentinel

A self-hosted AI gateway that masks personal data before it reaches an external model, and automatically classifies every document against GDPR and the EU AI Act — built as the DACH Enterprise AI Gateway, with a small public demo of both capabilities

Prepared September 2026 · Core gateway built and internally verified; public demo built and embedded in Envoy's live interface

1. Executive Summary

What it's for. A company that wants to use AI on its own documents — contracts, HR letters, customer records — faces a real conflict: those documents often contain personal data, and sending personal data to an external AI provider can violate privacy law (GDPR) or a company's own policy, especially for organizations operating in Germany, Austria, or Switzerland (the "DACH" region), where data-protection enforcement is strict. There is a second, quieter problem behind the first one: even before AI enters the picture, most companies have no automatic way to know, at the moment a document is stored, what regulatory category it falls into or who inside the company should even be allowed to access it. Data Sentinel exists to solve both — using AI on sensitive documents safely, and knowing automatically how a document should be classified and routed the moment it arrives.

What it is. Underneath the name "Data Sentinel" is a substantial piece of infrastructure Luis built and calls the DACH Enterprise AI Gateway: software a company runs on its own servers, sitting between its staff or systems and whichever AI provider it uses. It does two distinct things that reinforce each other. First, every document that passes through has personal information found and masked automatically before anything leaves the building, and restored back to the real values once the AI's answer comes back, so a user sees a normal, readable result while the outside AI provider never saw the real data at all. Second, the same underlying analysis can run a GDPR/EU AI Act compliance read on a document automatically as it's stored — not just as a one-off report, but as a classification a company can use to automatically determine the right processing path and the right access rights for that document, by department or seniority, before any person or downstream AI system touches it. A small, safe public demo of both ideas is built to be embedded on Luis's portfolio site.

What it does. In practice: a document goes in, personal data is found and masked before any AI model sees it, the request is routed to an appropriate AI model — a fast cloud model, a stronger one, or, if the company wants zero data leaving its network at all, a fully local one — a response comes back, the personal data is restored, and the company gets a clean, usable answer with a full audit trail. Separately, and just as usefully, the same document can be run through an automatic compliance classification the moment it's stored — producing a clear read of its regulatory sensitivity that a company can use to set who is allowed to see it and how it should be handled, including by whatever automated or agentic system processes it later.

2. Technical Summary

What it does, technically. The gateway sits in front of any outbound LLM call. Incoming documents are chunked, run through Presidio-based PII detection and anonymization — including custom-built recognizers tuned for German-language documents — and masked using reversible, encrypted tokens rather than simple blanking, so the same values can be decrypted back later. Each chunk is then routed through a model-agnostic proxy layer that can call a cloud model or a fully local model depending on how the deployment is configured, optionally grounded with retrieved context from a RAG pipeline, and the response is de-anonymized before it's returned. Separately from that per-query flow, the gateway's EU AI Act module can run a document through a compliance classification pass at ingestion time — a structured read of its regulatory category and sensitivity — which is the input a downstream system needs to assign the correct access rights and processing path automatically, rather than a person triaging every document by hand. Every step, on both paths, is traced and logged for audit purposes.

How it does it. The backend is a FastAPI service built around LangGraph (an agent-orchestration framework) for the tool-using logic, with a Next.js frontend. Outbound model calls go through a LiteLLM proxy configured with named model aliases — a fast EU-hosted tier, a stronger reasoning tier, a multilingual embedding model, and a fully local model for a zero-external-egress mode. PII masking runs through a pair of Presidio containers (analysis and anonymization); the compliance-classification path is governed by the gateway's own EU AI Act module, one of nine modules in its registry; n8n handles workflow automation; Redis provides caching; and in its fuller deployment profile, Postgres with pgvector and Langfuse add persistent storage and observability. What capabilities are actually available at runtime is governed by a real module registry — the system checks which optional pieces are actually configured and only advertises what genuinely works.

How it was built. The gateway was built from an architecture blueprint through a structured process: six build stages implementing different parts of the system in parallel, followed by a

hardening round that fixed defects surfaced during live end-to-end testing. The public demo came afterward, as a deliberately separate, much smaller build: it borrows only the masking and compliance-read ideas — a couple of pattern files were copied over by hand — not the gateway's code or runtime, and works against four fixed, pre-processed sample documents rather than live infrastructure, so it can be exposed safely on a public portfolio site.

3. Business-Scope Documentation

The problem it addresses

AI providers like Google or OpenAI are, from a data-protection standpoint, a third party. Sending them a document containing a customer's name, address, or contract terms means that third party now has that personal data — which under GDPR requires a legal basis and often explicit safeguards, and under the incoming EU AI Act carries additional obligations for anything resembling automated decision-making. Many companies in the DACH region want AI's productivity benefits but cannot accept that exposure. Separately, most companies also have no systematic, automatic way to classify a document's sensitivity and route access to it correctly the moment it arrives — that triage is usually manual, inconsistent, or simply skipped. Data Sentinel is built to resolve both problems together, since they share the same underlying analysis of a document's content.

Capability 1: masking for AI queries

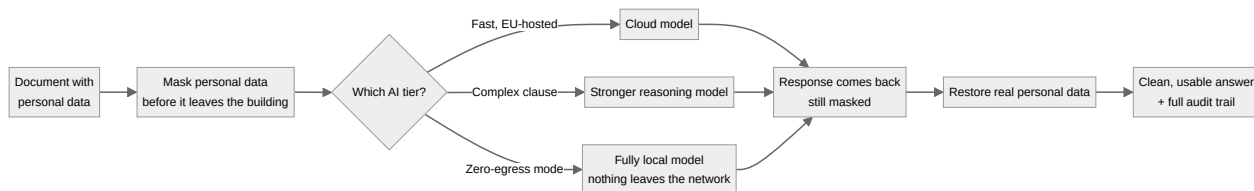


Fig. 1 — Capability 1: mask before an AI query leaves the building, restore before the user sees the answer.

Every request through this path produces a trace: what personal data was detected, what was actually sent to which model, and what came back. This is designed to answer, directly, the question a compliance officer or auditor will eventually ask: what left this building, and to whom. The zero-egress mode — routing to a fully local model instead of any cloud provider — exists for the strictest cases, where even masked data leaving the network is unacceptable.

Capability 2: automatic compliance classification and access control

This is the less obvious half of the system, and arguably the more broadly useful one: the same analysis that lets the gateway mask a document also lets it read that document against current GDPR and EU AI Act text and produce a structured compliance classification — automatically, at the moment the document is stored, not only on request. That classification is exactly the input a company needs to answer two operational questions it otherwise has to answer by hand for every document: what processing is this document allowed to go through, and who is allowed to access it. Once a document carries that classification, access and routing rules — by department, by seniority level, by document category — can be derived and applied automatically, including by whatever agentic or automated system handles the document downstream, without a person manually tagging or gatekeeping it first.

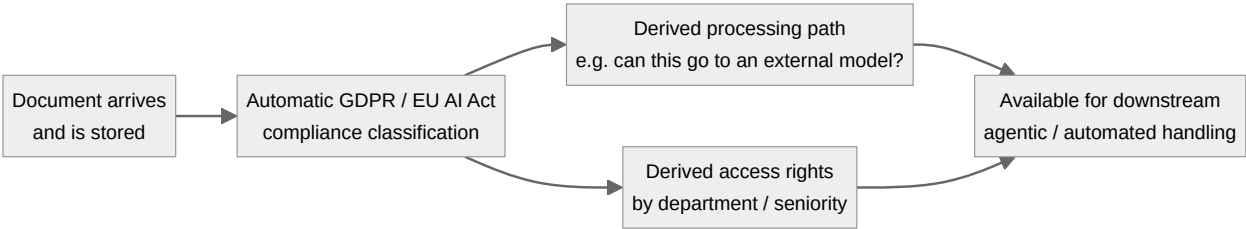


Fig. 2 — Capability 2: classification at intake becomes the basis for automatic routing and access control.

This matters for a different reason than the masking capability: it removes manual compliance triage as a bottleneck, at the point where documents enter a company's systems rather than only at the point where someone happens to query an AI about one. It's also the capability that makes the rest of the system's outputs usable inside a larger automated pipeline — an agentic system further downstream can trust the classification already attached to a document instead of re-deriving it.

A modular, pay-for-what-you-use model

The gateway is organized around a real module registry, not a single all-or-nothing deployment. The system checks at runtime what's actually configured and licensed, and degrades gracefully rather than breaking when something isn't present — a client can start with the core masking-and-routing capability and add modules as their needs grow, without redeploying the whole system.

MODULE	WHAT IT ADDS
Core analysis	Document chunking, PII masking, model routing — the baseline
Agentic router	Per-clause routing between fast and reasoning-tier models
RAG grounding	Answers grounded in retrieved precedent, not just the input document
HITL review	Human-in-the-loop checkpoints before a result is finalized

EU AI Act module	Automatic per-document compliance classification against GDPR/EU AI Act — the basis for automatically derived processing paths and access rights
Sovereign mode	Fully local model, zero external network egress
Observability	Full request tracing and monitoring
Connectors	Pulling documents in from external systems (e.g. a shared folder, ticketing system)

The public demo's role

The internal gateway itself is not something to expose on a personal website — it has no authentication layer by design, since it's meant to run behind a company's own private network. The public demo exists purely to let a portfolio visitor experience both capabilities safely, on four fixed sample documents (two in English, two in German): the masked-and-restored version of each document (Capability 1), and an independent compliance read of each, citing specific GDPR/EU AI Act articles (Capability 2) — deliberately shown as two equally real, equally demonstrated ideas, not one primary feature with a compliance footnote.

Integration status: Envoy's own frontend source code confirms this demo is wired in, not just built-and-ready — it's reverse-proxied by Envoy's server at `/data-sentinel/*` and embedded via an `iframe` on that same-origin path, requiring no separate public hostname or Cloudflare Tunnel route. It runs as a genuinely separate application (its own container, its own Next.js build) that shares no runtime with the internal gateway, reached entirely through Envoy's own interface.

4. Technical-Scope Documentation

Gateway architecture

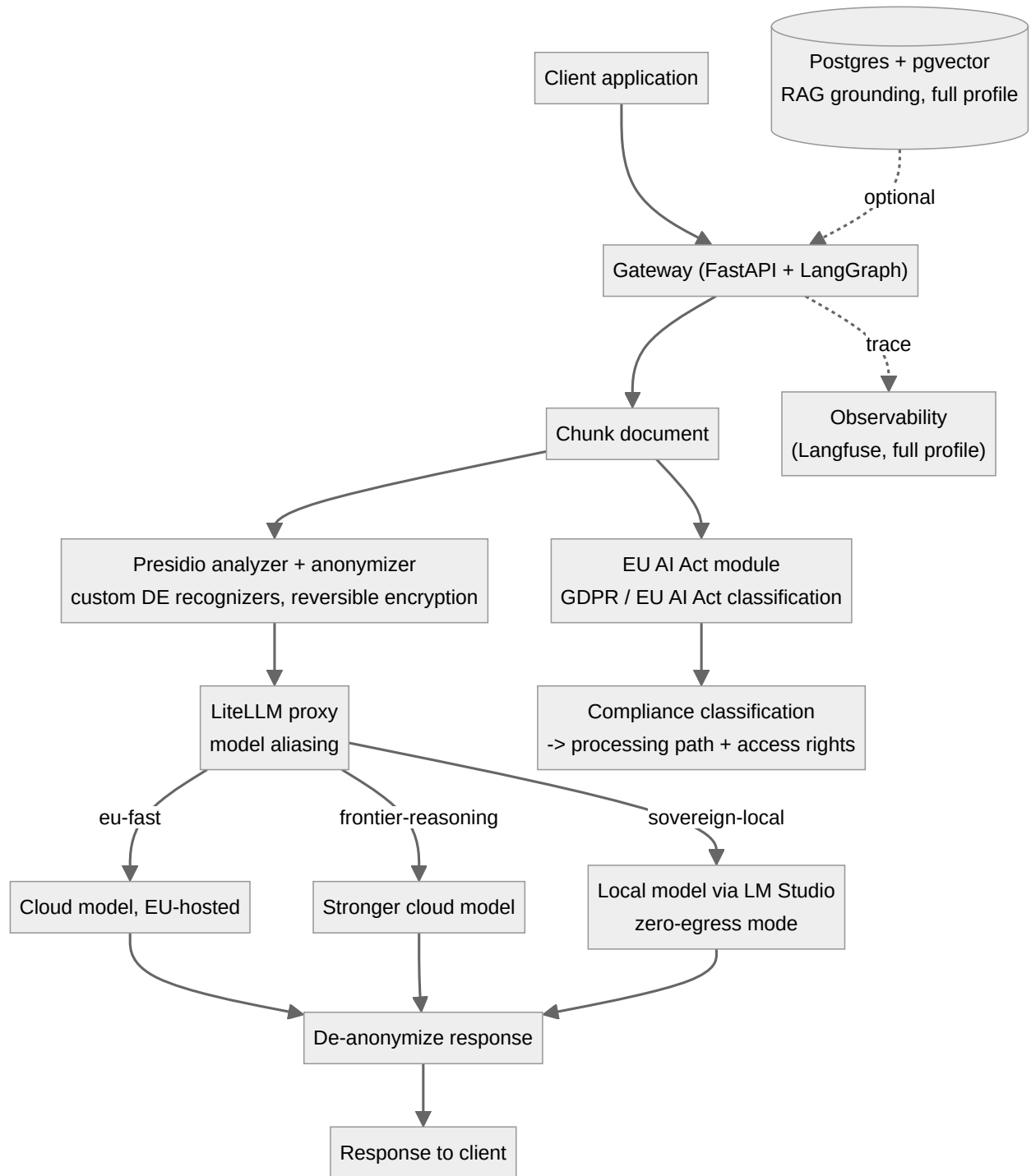


Fig. 3 — Gateway request flow: masking-and-routing on one path, compliance classification on the other, both fed by the same document analysis.

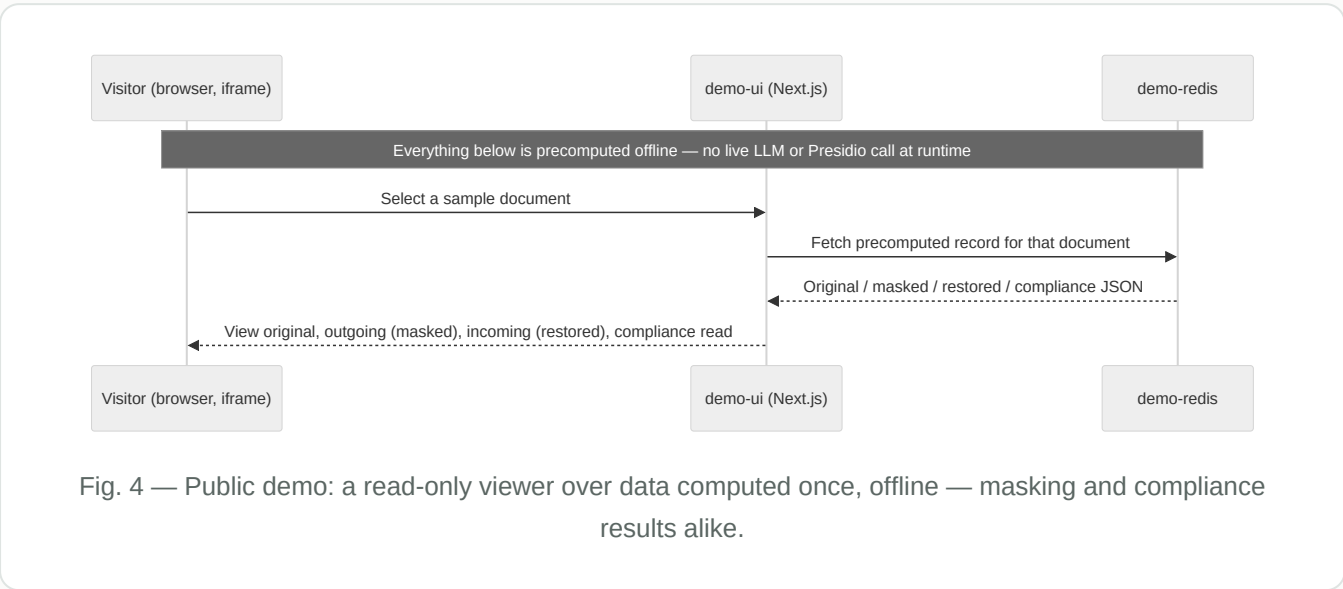
Confirmed service topology

SERVICE	ROLE
Gateway (FastAPI + LangGraph)	Core orchestration — chunking, tool-using agent, routing logic, module registry (including the EU AI Act classification module)

Presidio analyzer + anonymizer	PII detection and reversible masking, custom German recognizers
LiteLLM proxy	Model-agnostic routing, named aliases per tier
n8n	Workflow automation
Redis	Caching
Next.js frontend	User-facing interface
Postgres + pgvector, Langfuse (full profile)	Persistent storage and observability, optional deployment overlay

Base deployment is six backend services plus the frontend; a fuller profile overlays a database and observability stack on top. The module registry intersects a declared module manifest against which optional dependencies are actually configured at runtime, and exposes the result so the system only claims capabilities it genuinely has active — the EU AI Act module included.

Public demo architecture



The demo is intentionally a separate, minimal codebase: two containers (a password-protected Redis instance with no host port published, and the Next.js app), reading four pre-computed JSON payloads generated once, offline, by a standalone precompute script that is never built into the deployed image. There is no text input that reaches an LLM or Presidio at runtime — every visible result, masking and compliance alike, was generated ahead of time from a fixed, hand-chosen document set. This structurally rules out the misuse cases a public-facing demo would otherwise need to defend against, rather than relying on runtime filtering to catch them.

Embedding is designed around an iframe: the demo reports its own content height to the parent page via `postMessage` so the embedding site can resize the frame responsively, and a content-

security-policy setting restricts which origins are allowed to frame it at all — a closed-by-default gate rather than an open one.

State of verification

The gateway's own documentation is candid about where its "live-verified" evidence currently stands: an earlier round of end-to-end testing was run against a prior container-image configuration. Since then, the Presidio and LiteLLM services were switched to custom, slimmer container builds, and that switch has not yet been re-verified end-to-end against the same tests. This is a disclosed, self-flagged gap in the project's own build notes, not a discrepancy this review uncovered independently — worth carrying into any technical write-up as "last fully verified against a prior image set."

A source bundle of the gateway was generated on 2026-09-07, intended to be unpacked and built fresh on a deployment host (with real secrets, dependencies, and a corpus fetch left for that host to complete) — evidence the project was being staged for handoff or deployment around that date, not evidence of an already-running public instance.

5. Demo

Executive summary

Because the real Data Sentinel gateway is meant to run privately inside a company's own network — it has no login screen, by design, since it isn't meant to face the public internet — Luis built a separate, much smaller public demo that lets a portfolio visitor see both of its core ideas work without exposing any real infrastructure. A visitor picks one of four sample documents (two in English, two in German) and can step through what happens to it: the original, the version with personal data masked before it would leave the building, a simulated AI reply with that personal data restored, and — equally — an independent compliance read of the document against GDPR and EU AI Act rules, all real content, none of it faked for effect.

Technical summary

The demo is a deliberately minimal, self-contained build: a Next.js application and a password-protected Redis cache, nothing else. Everything a visitor sees was computed once, offline, by a standalone script that runs the real Presidio masking-and-restoring pipeline and the compliance-classification logic against the four fixed documents and writes the results to JSON; that script and its dependencies (Presidio, spaCy, any LLM key) are never included in the deployed image, so the always-on public containers never call an AI model or a PII engine at runtime — they only serve pre-computed results out of Redis.

Business and technical scope

The demo exists to solve a specific trust problem for two claims at once: "personal data gets masked before it leaves your environment," and "this document has been read against actual, current regulation, not a guess." Both are easy claims to make and hard to make credible from a paragraph of text alone. Letting a visitor click through the actual before/after on a real document, and read a compliance report that cites specific GDPR/EU AI Act articles, makes both mechanisms concrete rather than asserted — for an audience of recruiters, technical reviewers, and potential clients evaluating the underlying gateway who have good reason to be skeptical of unverified data-handling claims.

Technically, the project shares no runtime or code path with the internal gateway — a couple of PII-recognizer pattern files were copied over by hand, credited in the file that uses them, and that is the entire connection. It is embedded via iframe on Envoy, confirmed directly in Envoy's own frontend source: Envoy's server reverse-proxies `/data-sentinel/*` to this app's container and mounts it in an iframe pointed at that same-origin path, so no separate public hostname or Cloudflare Tunnel route is needed. It reports its own rendered height to the parent page automatically so the iframe resizes itself, and a content-security-policy setting restricts which origins are allowed to frame it at all, closed by default until the real site's origin is explicitly listed. The demo's containers, Redis-backed API routes, and all four documents' precomputed output — masking and compliance both — are complete, internally consistent, and wired into Envoy's live interface, not a pending integration.