Cogeto

How Cogeto makes the technical side of compliance demonstrable, not asserted

Compliance one-pager · Last updated 2026-08-01 · cogeto.eu

Cogeto turns an organisation's documents into verified institutional memory: every fact traced to its source, verified before it is stored, checked for contradictions against everything else held, and provable to a third party. Every trust claim is backed by an artifact you can inspect. This one-pager summarises where data lives, how it is isolated and protected, how deletion, export, and the signed findings report work, how subprocessors are determined, and how the design maps to EU regulation. Where a control is organisational or contractual rather than technical, we say so plainly. Architecture alone is not compliance: it is the part of compliance a system can prove.

Who we are

Organisational or contractual control

Cogeto is built and operated by MCTO Advisory d.o.o., incorporated in Croatia, in the European Union.

For a managed deployment, the customer is the data controller and the operating company acts as processor on the customer's behalf. For a self-hosted or fully offline deployment, the customer operates the instance itself and no operator processing takes place.

- **Roles:** Controller and processor roles are fixed per deployment in the data processing agreement.
- **Contact:** hi@cogeto.eu

Where data lives

Every deployment is a single isolated instance serving exactly one customer, brought up on EU infrastructure of the operator's choosing. There is one composition of application processes, relational database, vector index, object storage, and edge proxy per customer.

- **EU managed cloud:** Run on EU infrastructure under EU data terms.
- **Your servers:** Self-host the full stack on your own infrastructure.
- **Offline bubble:** Fully isolated, local models only, nothing leaves the box.
- **Model residency:** For a managed deployment the model provider is chosen with the customer and can be required to be European; the model layer can be pointed entirely at local models so external model traffic becomes zero.

Isolation model

Isolation is single-tenant by construction, not a configuration option. Because each customer runs a separate instance, there is no shared index, cache, or queue in which one customer's data could ever neighbour another's, and no aggregate store whose access control has to be trusted.

We state the residual risk plainly: a compromised host compromises its own instance, as in any single-tenant design. The blast radius of that event is one customer, which is precisely the point of the isolation model. Inside an instance, spaces partition content the same way: a space is a wall, not a filter, so separate mandates, clients, or departments never share retrieval.

- **No co-tenant:** One customer per instance; no cross-tenant data path exists to secure.
- **Space boundary:** Everything content-bearing belongs to exactly one space, and no feature operates across two. Deletion, receipts, and passports are per space.

Encryption in transit and at rest

In transit, transport security terminates at the instance's edge proxy, and every application endpoint requires authentication before it will answer.

At rest, original documents are encrypted in object storage and on the host's volumes, and connector credentials are additionally encrypted at the application layer before they are stored.

- **Key handling:** Object-storage keys are scoped by organisation, user, and visibility, and originals are reachable only through short-lived signed URLs gated by the sensitivity flag.

Verifiable deletion

A deletion receipt is a fileable, verifiable proof that a deletion happened and exactly what it removed. Deleting a source runs a five-step compensating protocol across the relational store, the vector index, and object storage, and ends in a signed receipt.

Each receipt records the source, the counts of memories, vector points, and bytes removed, and the pending and confirmed timestamps. It is signed with the instance key and chained to its predecessor, so no receipt can later be altered or excised without breaking the chain. A nightly sweep re-verifies that nothing deleted still exists, which upgrades each receipt from a record of intent into a continuously re-verified statement of fact.

> **Honest limitation.** Receipts prove removal from the system's own stores. They cannot retroactively unsend content already processed by an external model provider under that provider's terms, which is exactly why deletion is paired with the local-model isolation option that bounds what leaves the instance in the first place.

A sample deletion receipt


```
{
  "id": "55555555-5555-4555-8555-555555555555",
  "source_type": "user_note",
  "source_id": "note-old",
  "counts_json": {
    "requested_by": "ana",
    "enumerated_at": "2026-06-30T12:00:00.000Z",
    "memory_count": 2,
    "object_keys": []
  },
  "signed_at": "2026-06-30T12:00:03.000Z",
  "confirmed_at": "2026-06-30T12:00:03.000Z",
  "prev_hash": "cogeto:deletion-receipt-chain:genesis",
  "hash": "0000000000000000000000000000000000000000000000000000000000000000",
  "signature": "c2FtcGxlLXBsYWNlaG9sZGVyLXNpZ25hdHVyZS1ub3QtcmVhbA=="
}
```


Illustrative only. This is the fictional sample from the public Memory Passport (a demo persona in the sandbox). Its hash and signature are placeholders, not real cryptography. A receipt from a live instance carries a real hash chain and a real signature you can verify against the instance public key.

Data export: the Memory Passport

There is no lock-in. Any user can export everything they hold, from facts and their statuses to sources, full validity history, and deletion receipts, as a single signed archive in an open, documented, versioned format.

The format is published in full. A Passport can be read and its integrity verified with only the public schemas and the instance public key inside the archive, with no Cogeto code or service required.

- **Open format:** A single .zip with per-document JSON, each covered by a public JSON Schema, plus a detached signature over the manifest.

The signed findings report

A findings report is a signed, printable record produced over an explicitly selected set of sources. Its header states the instance, the corpus scope, the date range, the model configuration, and that configuration's measured trust scores, so the artifact declares the accuracy of the system that produced it.

Each contradiction appears with both claims, both verbatim source sentences, the document with revision and location for each side, the detection date, and the resolution status. Superseded statements appear with their chains, and a summary of what verification rejected follows. Findings that reference a document outside the selected scope appear in a clearly labelled boundary section rather than being silently included or dropped.

- **Two formats:** PDF for people, JSON for quality systems, both signed through the same path as deletion receipts.
- **Defined scope:** The signature covers an explicitly selected document set, so what was audited is part of the record.

How Cogeto receives email

Inbound email is disabled by default and enabled per deployment only when the customer wants it. When enabled, email enters Cogeto by forwarding only: the deployment exposes a dedicated inbound address, and the user or their mail system forwards messages to it. Cogeto never connects to a mailbox.

A sender allowlist controls which senders may forward material in; messages from senders that are not on the allowlist are not ingested. The channel is receive-only: there is no inbox access, no IMAP connection, and no mailbox OAuth scope, and Cogeto cannot send, delete, or modify anything in the user's mailbox.

- **Off by default:** The inbound email service runs only when explicitly enabled for the deployment.
- **Receive-only:** Forwarding in is the only path; nothing reads or writes the source mailbox.
- **Sender allowlist:** Only approved senders can forward material into memory.

Subprocessors

Organisational or contractual control

The subprocessor list depends on the deployment. A self-hosted or fully offline instance may have no external subprocessors beyond the customer's own infrastructure. The subprocessors engaged for a given deployment are named in that deployment's data processing agreement.

- **Model and embedding inference:** The model provider the operator configures in the interface after install. Cogeto ships with none preselected and holds no model key of its own. In a fully isolated deployment no external model provider is used at all.
- **Hosting and infrastructure:** The EU cloud or on-premise infrastructure chosen by the operator for the single-tenant instance.

Regulatory mapping

We do not claim that architecture alone constitutes compliance, which always also has organisational and contractual components. We claim that this architecture makes the technical share demonstrable rather than asserted.

- **GDPR Article 17, right to erasure:** The deletion protocol and its hash-chained receipts give the data subject not only the deletion but the evidence of it.
- **GDPR Article 25, data protection by design and by default:** Single-tenant isolation, mandatory

provenance, hard visibility gates evaluated inside the store, encryption at rest, and minimal external data flow are properties of the architecture, not settings on top of it.

- **AI Act, transparency posture:** The system is built to show its work: a source link on every fact, an explicit status on every belief, an audit record on every action, published prompts, and published accuracy on the trust-score page.

Live trust score: cogeto.eu/trust

Every claim on this page corresponds to a mechanism in the product. Read the whitepaper for the full design and the trust-score page for measured accuracy, and your security team gets the design documents in full during review.

Whitepaper: cogeto.eu/documents/cogeto-whitepaper.pdf Trust scores: cogeto.eu/trust