From Regulation to Evidence

Building Compliance on Nick



Mack Palomäki · Ljusa Tider



Fifteen years.

Why? For what?

From birth until our final breath, we comply.

From birth until our final breath, we comply.

A regulator can ask you to prove it.

The auditor never asks whether it was recorded. The auditor asks: show me why this control exists, and prove that it worked.



2 a.m.



Thirteen countries. Thirteen first languages. Every instruction understood by all of them, at three in the morning.

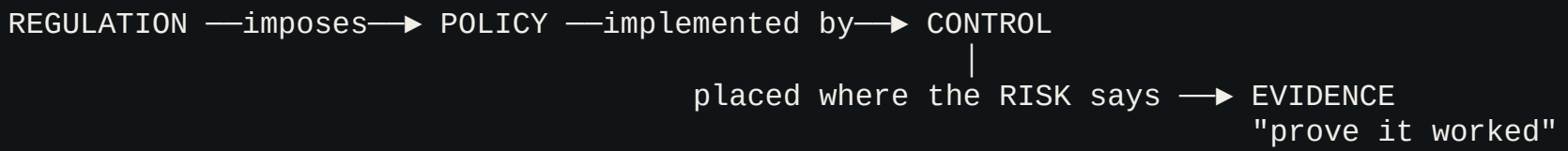
Six hand-overs between seven systems that never talked to each other. I became the integration layer, every night, by hand.



A compliance system needs five things.

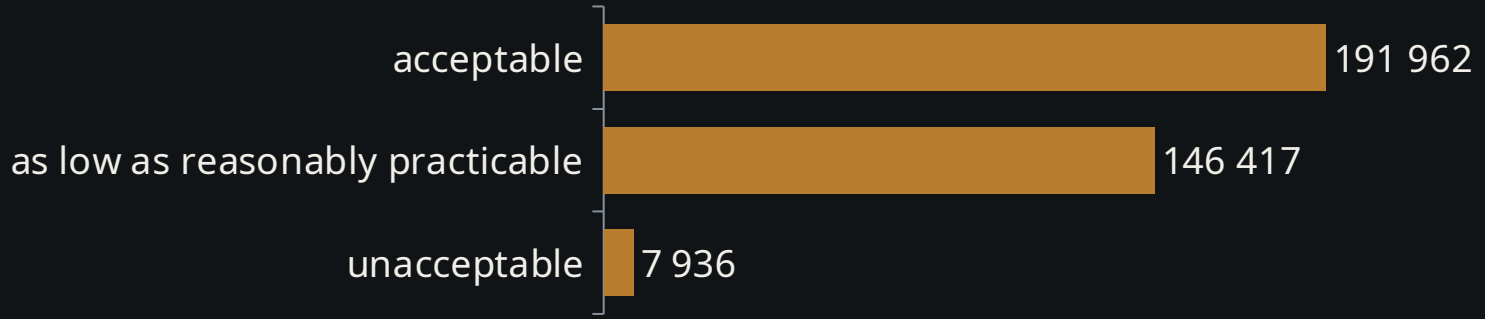
1. A chain an auditor can walk
2. A unit smaller than the document
3. A register with workflow, versions and roles
4. Links that mean something, and evidence nobody can edit
5. An audit trail nobody can rewrite

1. A chain an auditor can walk: regulation → policy → control → risk → evidence. Break one link and there is no answer, however many documents you have.



1. A chain an auditor can walk: regulation → policy → control → risk → evidence. Break one link and there is no answer, however many documents you have.

The risk link is not a document either. Every source message carries a severity class, every translation a risk number, and 7,936 of them are flagged unacceptable by name.

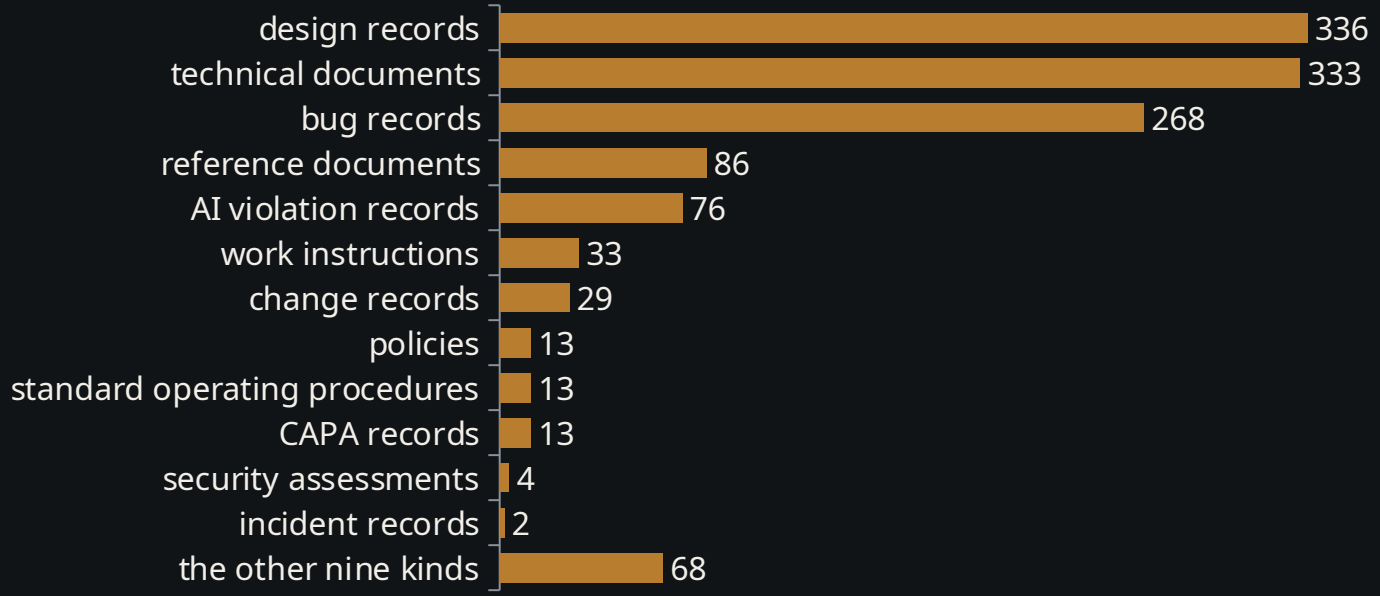


2. The unit is not the document. It is the claim inside it: who says so, what backs it, who checked it, until when.

```
CLAIM      "This plywood is free from SVHC substances above 0.1 weight %"
  made by         a supplier
  evidenced by    a certificate, dated
  evaluated by    a named auditor, on a date
  result          meets requirement
  valid until     the certificate's expiry
```

3. A register with workflow, versions and roles. Plone has had this for twenty years; Nick carries it forward.

3. A register with workflow, versions and roles. Plone has had this for twenty years; Nick carries it forward.



4. Links that mean something, and evidence that cannot be edited afterwards. This is what I added.

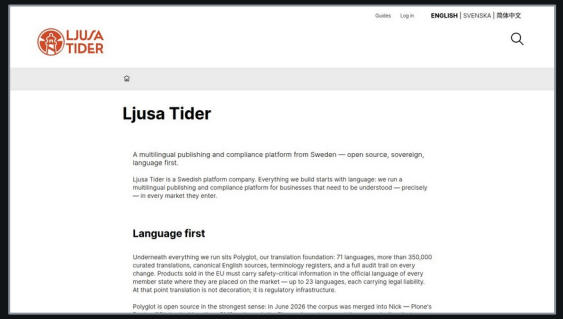
qms.controlled_documents	1,274 docs	26 document types · 12 domains
qms.document_links	64 rows	19 relationship types, each with its inverse
qms.document_evidence	178 rows	8 evidence types, phase-gated, immutable
qms.document_transitions	2,289 rows	every status and version change
qms.failure_modes	48 rows	310 occurrences · 261 standing controls
qms.training_*		modules → steps → sessions → certificates

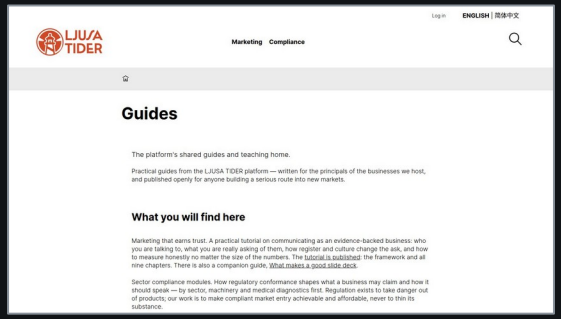
5. An audit trail nobody can rewrite. PostgreSQL underneath: 197 triggers, 49 of them writing insert-only audit tables. The database is the audit surface.

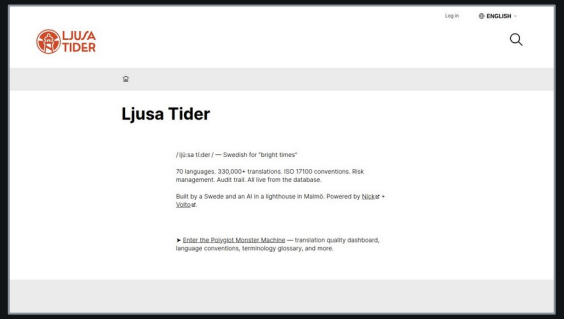
Why Nick: Plone's model, one language front to back, PostgreSQL underneath, and a codebase small enough that one person has read it whole. An AI works well on it for the same reason.

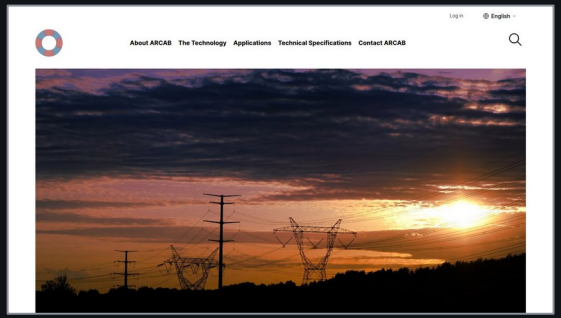


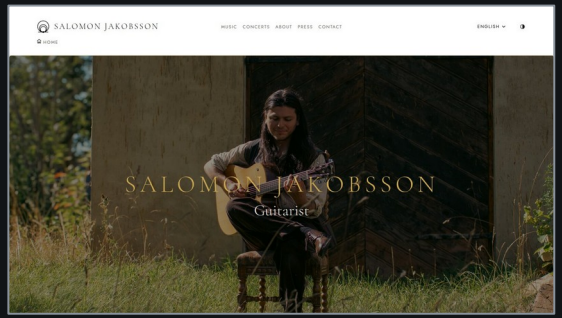
It runs: eleven sites on one host, one database each, the platform's own QMS among them, every tenant through the same code and the same tools.



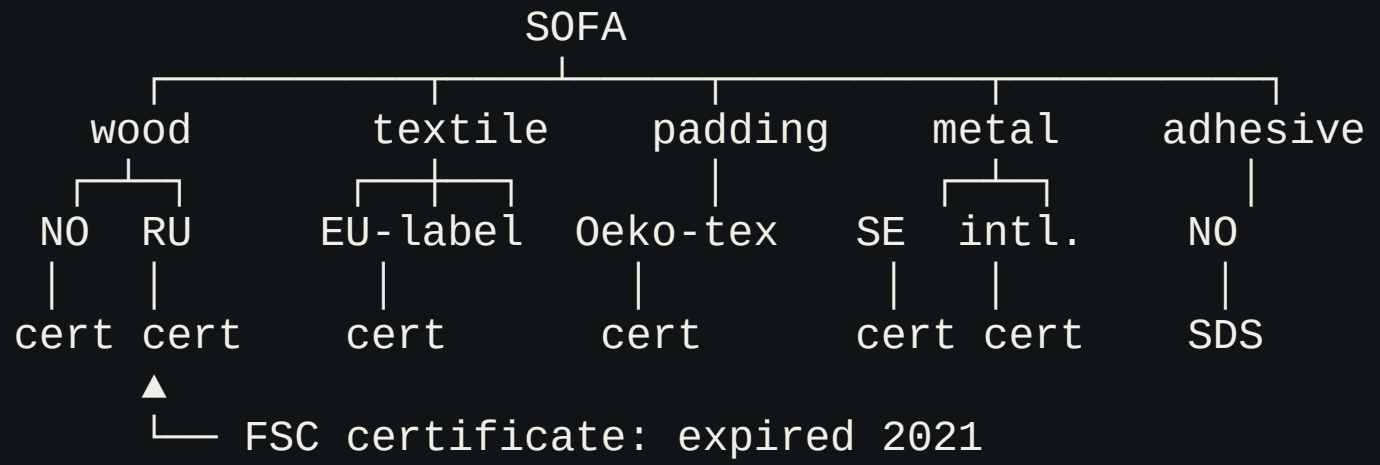






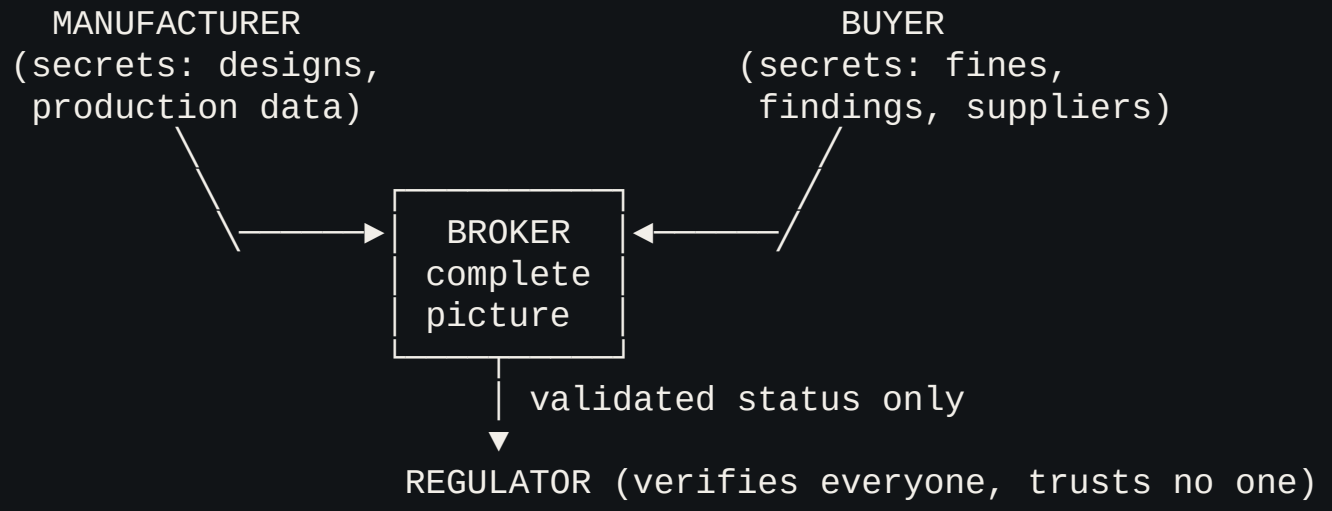


One sofa, thirteen suppliers, four countries, and one forestry certificate that expired in 2021 while nobody was watching.

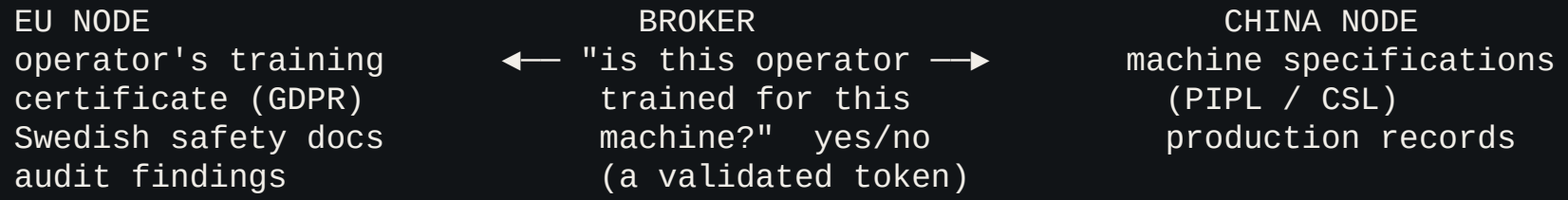


A machine arrived without CE marking. The buyer was fined, did the compliance work, and became the manufacturer in law. The role decided the identity.

Compliance data is where an organisation keeps its fines. So the platform holds the complete picture and no single participant sees everything: visibility follows relationship and role, never folder.



The compliance question crosses the border. The data does not. That is a federation, and you cannot decide to become one late.



Every language model runs on my own hardware, never on the production server. An AI endpoint on a regulated system is a significant change, so the models stay outside and only their results cross, through change control.

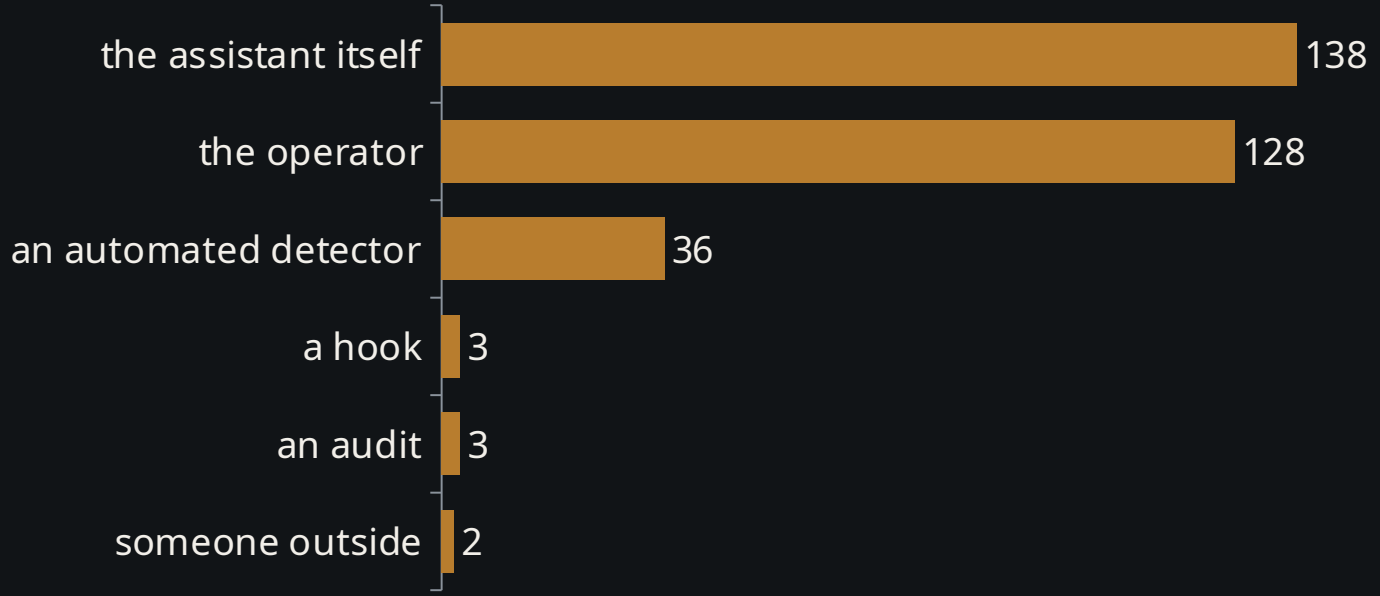
Repositories holding audit findings are themselves regulated material. So the code lives on our own git server, inside the EU.

The AI is good at Nick for the reason an auditor is: it can read all of it before it acts. The property that lets one person audit a codebase is the property that lets an assistant work on it well.

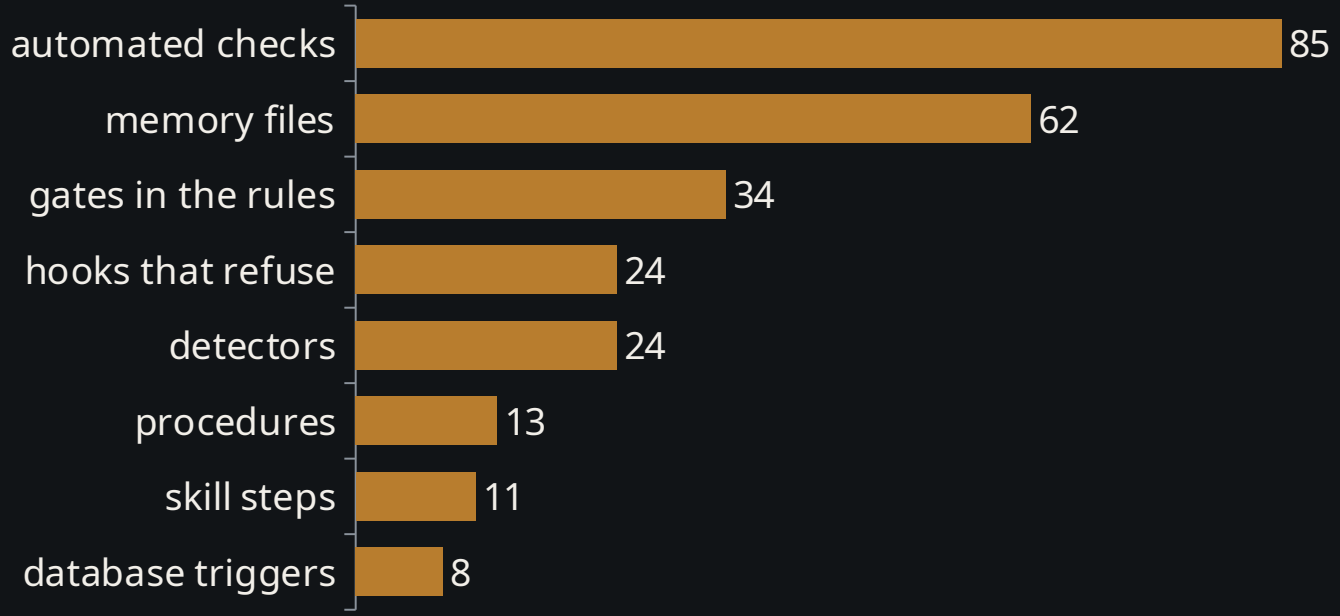
"AI must be held to the same discipline as human developers — everything auditable, every change documented before deployment."

76 recorded AI violations, 48 named failure modes, 310 occurrences, and a record of who caught each one.

76 recorded AI violations, 48 named failure modes, 310 occurrences, and a record of who caught each one.



261 standing controls, each tied to the failure mode it exists for. Not better prompts: a register of how it fails, and a control for each way.



Eighteen hours a day became six, and throughput went up. The rails are not bureaucracy around the AI. The rails are the trust.

380 effective, 657 in draft, one training module, and the interface a quality manager could use without me is the least finished part.

1 Compliance is proving, on demand, that a rule was met. Proof is records that point at each other.

2 The unit is the claim, not the document: who says so, what backs it, who checked it, until when.

3 Nick is a system for records that point at each other: Plone's model, PostgreSQL underneath, small enough for a person and an AI to read whole.

4 That is the way out of the cost of over-regulation: proved once, watched for you, shown to whoever asks. Compliance stops being a tax and becomes useful.

5 An AI near regulated work needs a register of how it fails, and a control for each way.

- Where should a claim live in Plone: a content type, a relation, or a table beside the content?
- When the structure is machine-made end to end, what does the human signature mean?
- Who else here is building compliance on Plone or Nick?