19

THE STACK BEHIND THE AI COWORKER

Compliance Is Not a PDF

Dr Peter McCann Strain, CTO and senior AI engineer, DPhil/PhD in AI from Oxford University

A regulator wants one customer's hour reconstructed by Friday. A thumbs-up emoji won't do it.

An essay in the series **Architecting the AI Coworker**.
Approx. 26 minute read · Essay 19 of 22

Dr Peter McCann Strain
CTO, DPhil/PhD in AI from Oxford University

> A regulator emails on a Wednesday. Call her Ms Adeyemi, because the request is going to come back, and a request with a name attached is harder to file under "later". Her tone is cordial, the way a tax inspector is cordial. She would like to know what your agent did with one specific customer's data between two o'clock and three o'clock on the afternoon of 17 March. Not a summary. Not a policy document. The actions: the reads, the writes, the calls outward to other systems, the prompts that were assembled, the tools that were invoked, the humans who approved or did not, the model version that was running, the policy version that was live, and, because she is thorough, whether anything the agent learned that hour still lives somewhere in its memory.

You forward the email to the team. The team forwards it to the vendor. The vendor sends back a screenshot of a dashboard. The dashboard says "Conversation 8,412, closed successfully." There is a thumbs-up icon. There is no log of the tools the agent called. There is no record of the customer record it opened, modified, and closed again. There is a chat transcript, but it has been redacted by the vendor's own pipeline for "PII safety," which is darkly funny given that the question is, precisely, what happened to one person's PII. You stare at the screenshot for a long time. A thumbs-up is not an answer to Ms Adeyemi. It is an emoji.

The previous essay in this series argued that when an agent acts, the system must be able to say who acted: which human authorised it, which platform accepted it, what tool carried it out, who owns the consequence. An agent, in this series, is software that takes a goal, breaks it into steps, and acts on the world: it reads files, calls other software, sends messages, changes records. Identity answers who. This essay is about the question that follows the moment a regulator, a court, or an angry customer arrives: can the system prove what it actually did? Because compliance, in the agent era, is not a PDF you generate at audit time. It is an evidence trail the system produces continuously, as a side effect of working.

— FIGURE E19.1 · E19 - COMPLIANCE IS NOT A PDF

Twelve Runtime Compliance Primitives

An engineering synthesis, not a statutory form. Each row names one primitive; each primitive maps onto governance machinery the organization already runs.

	BUCKET	PRIMITIVE	WHAT IT PRODUCES	EXISTING MACHINERY IT EXTENDS
P1	Registries	Agent registry	Identity, owner, purpose, version of every running agent.	Asset inventory.
P2	Registries	Tool registry	What each tool connects to, permitted operations, owner.	API catalogue.
P3	Registries	Versioning record	Model build, prompt hash, policy and tool versions in play.	Change-management register.
P4	Authority and policy	Permission and mandate registry	Each grant: scope, subject, expiry, revocation path.	Identity and access management.
P5	Authority and policy	Purpose and policy record	Declared purpose, live policy version, forbidden-action tags.	Data protection impact assessment.
P6	Authority and policy	Risk-tiering record	Each action class graded by stakes, reversibility, observability.	Operational risk register.
P7	Runtime records	Real-time data-flow inventory	Every read, write, export, store; personal-data and cross-border flags.	Data lineage.
P8	Runtime records	Action ledger	Per action: timestamp, trace ID, tool, inputs, outputs, result.	Audit logs.
P9	Runtime records	Human approval log	What each approver saw and decided, reason and time taken.	Workflow approvals.
P10	Runtime records	Evaluation records	Test suites, failure classes probed, release-gate evidence.	Evaluation pipeline.
P11	Reversal and audit	Rollback and revocation record	Every undo, kill switch firing, memory-forgetting receipt.	Incident response.
P12	Reversal and audit	Audit-ready trace store	The join that turns the other eleven into one answer a regulator can read.	Audit-evidence function.

Run the twelve primitives against one deployed agent and one customer-hour; the gaps are the work.

Source: Authored as a synthesis of GDPR Art. 22, EU AI Act Art. 13 and 26, NIST AI RMF GenAI profile, and the OWASP LLM Top 10.

REFERENCE CHECKLIST · not a formal standard ARCHITECTING THE AI COWORKER

Twelve runtime governance primitives in four buckets: registries, authority and policy, runtime records, and reversal and audit. The audit-ready trace store is the join that turns the other eleven into an answer a regulator can read.

Agents break the bargain that let governance be a document

For most of the software era, governance was a description. The compliance binder of the SaaS years, the audit report on a provider's controls, the formal write-up of how a system handles personal data, the policy PDF with an executive's signature on it, was always a trailing

artefact. Each one describes intended control. And that worked tolerably well, because the system being described was mostly deterministic: it did the same thing every time, so you could read the code, the change logs and the access records and produce a credible document about it. The PDF was a simplification of something fundamentally legible.

Agents break that bargain. An agent retrieves a document, decides it is relevant, decides what to do with it, calls a tool, evaluates the result, asks a human or skips a human, writes something to memory, revises its plan, and sometimes acts in ways nobody enumerated in the policy PDF at all. The system you wrote the PDF about is not necessarily the system that ran at 14:37 on 17 March, when one customer's data passed through it. In deterministic software, governance could be a description. In agentic software, governance has to be a recording. No human description can keep pace with a system that re-decides its own behaviour at runtime.

That shift is why three kinds of evidence have to be kept rigorously apart, and why most teams quietly conflate them. A reasoning trace is the model's own narration of its decision process. It can be useful, but it can also be incomplete, strategic, or written after the fact to rationalise a decision the model reached some other way. It is a clue, not proof. A system audit trail is what the harness, the scaffolding that actually runs the model, records outside the model's control: tool calls, timestamps, credentials, approvals, file diffs, API responses, policy decisions, model versions, prompt versions, trace identifiers, data-flow records, and memory writes. This is the evidence layer most teams underbuild, and it is the one that matters. An external check is what something independent concludes on top of those records: a deterministic verifier running a fixed rule, a structurally different evaluator, a human reviewer, an auditor, a court, a regulator. The agent may explain itself, and the harness must evidence it, but neither the agent nor the harness should be the sole judge of that evidence.

That separation is also the answer to the most hostile read of the whole instrument: that the system is grading its own homework, producing the very trace that is supposed to exonerate it. The trace store's credibility cannot come from the agent's honesty, because the agent is exactly what is under suspicion. It comes instead from the agent's inability to alter the record after the fact, the same way logs become admissible elsewhere: append-only or write-once storage, spans cryptographically signed at the moment they are written, independent timestamping, and the external-check layer sitting above, a deterministic verifier the agent does not control. A record the suspect cannot edit is not the suspect's testimony.

The law has begun to make this distinction operational, and from more than one direction. Europe's AI rules now place on high-risk systems a set of duties that read like an engineering specification: automatic event recording that runs over the system's whole lifetime so its operation can be traced; genuine human oversight, a person who can understand, intervene in and stop the system rather than rubber-stamp it; and around those, a quality-management system, log retention by both the maker and the operator, post-market monitoring, serious-incident reporting, and meaningful explanations for certain decisions [1]. California pushes from the side of the courtroom rather than the regulator. From 1 January 2026, in a covered civil action against a defendant who developed, modified, or used AI that allegedly caused harm, the de-

fendant can no longer hide behind the claim that the AI autonomously caused it, although ordinary defences such as causation and foreseeability remain available [2]. The legal vocabularies differ. The engineering conclusion does not. If the action mattered, the system has to be able to reconstruct it.

The evidence the questioner wants does not change shape across regimes; only the letterhead does. This is the artefact the rest of the essay turns on, so let me state it precisely. Ms Adeyemi's request reduces to a single query, one the system either can or cannot answer:

> For customer `C-1842`, between `2026-03-17T14:00:00Z` and `2026-03-17T15:00:00Z`, show every action by `agent_id=support-refund-agent-prod`: every data read and write, every tool call, every human approval, every memory update, the model version and the policy version that were live, and any external message sent.

That query is the bar the whole essay is measured against. It asks for every action on one named person in one named hour, and it asks for the action in full: not just that the agent did something, but what it read, what it changed, what it called, who approved it, what it remembered, which version of the system acted, and whether anything left the building. A usable response to it does not need poetry. It needs joins, the ability to link separate records into one coherent account.

FIELD	EXAMPLE ANSWER	PRIMITIVE #	WHY IT MATTERS
Agent and owner	`support-refund-agent-prod`; owner: Head of Support Ops; version `2026.03.12`.	1, 2	Prevents orphan automation nobody owns.
Model and policy	Model build, system prompt hash, refund policy `v7.3`, data-minimisation policy `v4.1`.	3, 5	Explains which system actually acted.
Data reads	CRM profile, order `O-9912`, last three ticket summaries, loyalty tier.	7	Shows the personal data accessed.
Data writes	Ticket status changed to `pending_review`; draft refund note stored; no CRM field overwritten.	7	Separates read-only work from state change.

FIELD	EXAMPLE ANSWER	PRIMITIVE #	WHY IT MATTERS
Tool calls	`crm.get_order`, `policy.retrieve`, `refund.quote`, `email.draft`; no `refund.issue`.	8	Shows the action surface and the authority that was blocked.
Approvals	Refund over threshold routed to a human reviewer, who rejected autonomous issue and approved a draft email.	9	Makes oversight inspectable.
Memory writes	Temporary working summary expired after 24 hours; persistent preference memory unchanged.	11	Answers whether the hour still lives in memory.
External output	Draft email created, not sent; the customer received no outbound message in the hour.	7	Shows whether data crossed the boundary.
Trace and artefacts	Trace ID links to spans, tool payload references, the approval screen, the final ticket.	12	Lets an auditor replay the record without trusting narration.

The table above is a reference example, not a statutory form, but it is the operational bar. The rightmost column makes the relationship to the twelve primitives explicit: the nine rows of Ms Adeyemi's answer roll up to seven of the twelve, and the audit-ready trace store (Primitive 12) is the join that lets a regulator read all of them as one record. If the system cannot answer this query for one customer, one hour and one agent, then the PDF is doing far more work than the evidence, and the PDF will not survive contact with the regulator.

The clean case is the least interesting one. The instrument earns its keep when a row cannot be filled. Suppose the Memory writes row had come back not as "persistent preference memory unchanged" but as "UNKNOWN: the vector store has no per-customer key, so the system cannot confirm whether the hour still lives in memory." That single UNKNOWN is the line Ms Adeyemi escalates on, and it is also where the table does its real work, because it localises the gap to primitive 11 rather than leaving the whole answer in doubt. A success table is a record; a table with one honest UNKNOWN in it is a diagnosis.

Replace the policy PDF with twelve things the system must produce as it runs

If the policy PDF is the wrong artefact, what is the right one? Not a better PDF. The right artefact is the system itself, running, producing a small number of things continuously, the way an aircraft produces a flight recorder track without anyone deciding each second to write a line. I count twelve such things. A leader can argue with the number; the punch list is what matters.

One honest caveat before the list. The twelve are an engineering synthesis, not anyone's statute. They pull together obligations and risks already visible across Europe's AI rules, data-protection guidance, the emerging agent-identity and payment standards, California's liability posture and current legal analysis of agents under EU law. Others slice the same territory differently. Lin and colleagues' memory survey, for one, organises it around a six-phase lifecycle [11], and a practitioner who reads both will reasonably ask whether they are tracking twelve things or six. The mapping is not mysterious: Lin's six phases, write, store, retrieve, execute, share, and forget or roll back, are the lifecycle of a single primitive, number 11, viewed in motion; the other eleven are the apparatus that makes those six phases produce evidence. A lifecycle and a punch list answer different questions. Theirs is "what happens to a memory?" Mine is "what must the system emit so you can prove it?" The overlap is the topic, not the schema.

Run the list against one deployed agent and the output is a punch list rather than a maturity score, which makes it more useful, because a punch list is the one artefact you cannot frame and hang on the wall.

The twelve fall into four buckets, and the buckets are the point: four is a number a leader can hold in a meeting, and twelve is not. The four are registries, authority and policy, runtime records, and reversal and audit. Each answers one plain question about the system. What is running? What was it allowed to do? What did it actually do? And can any of that be undone and proven?

The Twelve Runtime Primitives

Registries: what is running

1. **Agent registry.** Identity, owner, purpose, version, deployment context and on/off state for every agent.
2. **Tool registry.** Connections, data classes, allowed operations, failure modes and owner for every tool an agent can call.
3. **Versioning record.** Model build, prompt hash, policy version, tool version and retrieval corpus in play at action time.

Authority and policy: what it was allowed to do

1. **Permission and mandate registry.** Scope, subject, expiry, approver, revocation path and kill switch for each grant of authority.

2. **Purpose and policy record.** Declared purpose, live policy version, data-minimisation envelope and forbidden-action tags by action class.
3. **Risk-tiering record.** Stakes, reversibility, observability and reach graded per action class.

Runtime records: what it actually did

1. **Real-time data-flow inventory.** Every read, write, export, store and onward recipient, flagging personal data and cross-border transfers.
2. **Action ledger.** Timestamp, trace ID, initiator, tool, inputs, outputs, policy check, result and corrective follow-up for every action.
3. **Human approval log.** What each approver saw, who they were, accept/reject/override, the reason and the time taken.
4. **Evaluation records.** Pre-deployment and continuous test suites, versions, failure classes, regression history and release gate.

Reversal and audit: whether any of it can be undone and proven

1. **Rollback, revocation and memory-forgetting record.** Reversal receipts, revoked permissions, memory-deletion events and the verification probe that confirmed each took effect.
2. **Audit-ready trace store.** A single joined, queryable store that links the other eleven into one account a regulator can read.

Twelve outputs, four buckets, one question: can the system reconstruct one customer's hour from records, not memory?

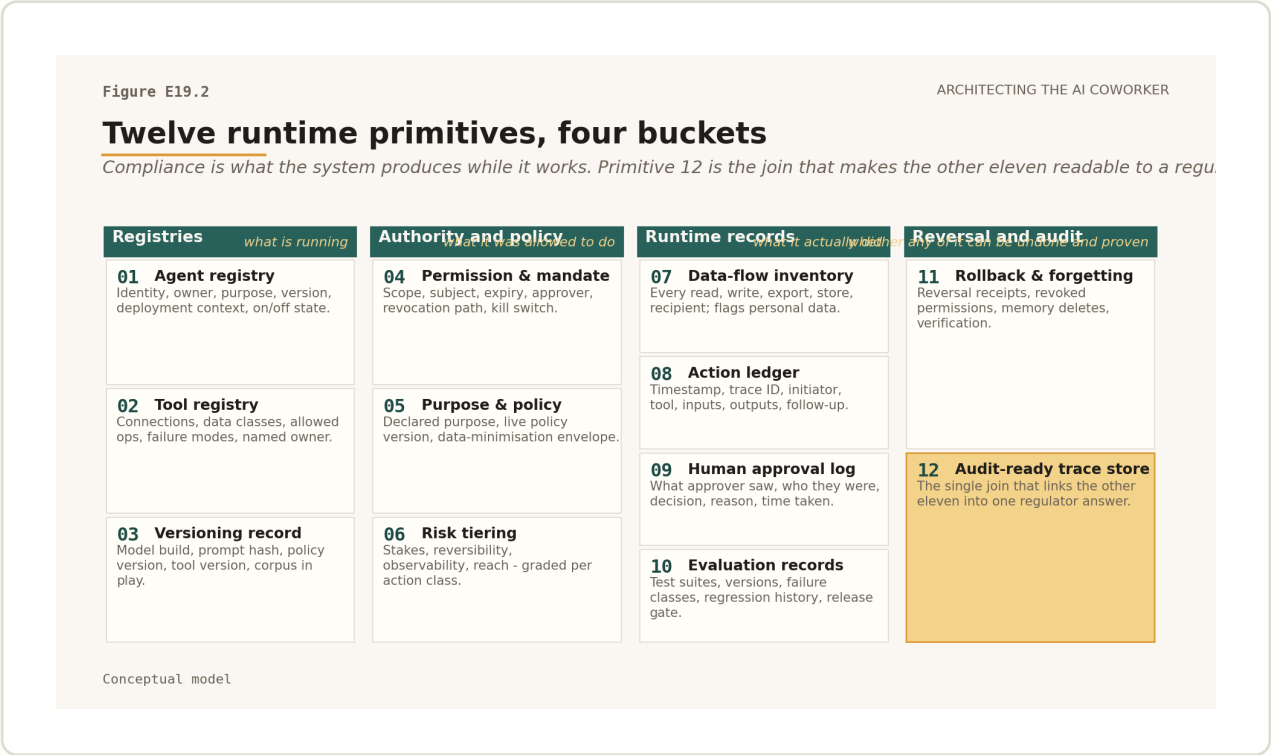


Figure E19.2. The twelve primitives laid out as a four-bucket grid, each cell carrying its one-line definition. Primitive 12, tinted amber, is the join that makes the other eleven readable as one regulator answer.

Do not read the twelve twice; the box is the list, and the prose has a different job, which is to make the list answer Ms Adeyemi. She needs four things, in order, and the buckets supply them. The registry layer tells her what was running, the agent, tool and version records, so no automation can disappear behind a product name. The authority layer tells her what it was allowed to do, the mandates, policy, purpose and risk tier, so a copilot label cannot hide a refund authority or a data export. The runtime layer is the record of what it actually did, the data-flow inventory, the action ledger, the human approval log and the evaluation records, so the answer is a sequence of events rather than an after-dinner reconstruction. And the reversal layer says whether any of it can be undone and proven, the rollback, the revocation, the memory-forgetting receipts and the joined trace store that turns raw logs into an account a regulator can read.

Most of this is not exotic. It is asset inventory, change management, identity and access management, privacy operations, data lineage, evaluation and incident response, given a new name for a new tool but doing the same old job. What is new is the demand that the agent produce all of it itself, as a side effect of acting, rather than a human reconstructing it afterwards from memory and hope. The hard part is not inventing the departments. The hard part is making them meet at the instant the agent acts, and the trace store is what makes them meet: a pile of logs is raw material, and the join is the thing that can answer Ms Adeyemi by Friday rather than by litigation. Notice, too, what is not on that list. There is no entry for the vendor's safety narrative, no entry for the executive's signed AI policy, no entry for a dashboard tile that

says "resolved." Those artefacts may matter in their own way. They simply do not substitute for the record of what happened.

A structural critic will press here: several of the twelve overlap, and the count could be collapsed to seven or eight without loss. The action ledger (8), the data-flow inventory (7) and the audit-ready trace store (12) all touch the same runtime events. The agent registry (1), the tool registry (2) and the permission and mandate registry (4) all describe what is configured to run. Why keep them separate? Because they answer different questions and have different owners, and a single mega-table collapses the questions before the team can use them. The action ledger answers "what did the agent do, step by ordered step?" It is keyed on action and is the natural artefact of the runtime team. The data-flow inventory answers "which personal data crossed which boundary?" It is keyed on data and is the natural artefact of the privacy team. The trace store answers "can these be joined into one account a regulator can read?" It is keyed on trace ID and is the natural artefact of the platform team.

Merge them and one team owns three jobs, and an outage in any of the three blocks all of them. Keep them separate, with the trace store as the join, and the same evidence is produced in three places that fail independently. The overlap is real; the separation is the point.

> **The PDF-vs-Runtime Test (digest of the twelve)**
>
> These four are not a fifth scheme. They are the four-line digest of the twelve primitives above, the test a leader can run on a Friday afternoon when there is no time to read all twelve. Compliance is not a PDF when these are produced as the system runs, not assembled afterwards from memory and hope:
>
> 1. **Per-claim provenance** (twelve-primitive joins: 7, 8, 12). Every assertion in the compliance answer carries a signed, timestamped trace identifier that resolves to the underlying record.
> 2. **Authority chain** (twelve-primitive joins: 4, 5, 9). For each action, the record shows who could permit it, what scope was granted, and which approval (human or policy) actually fired.
> 3. **Decision trace** (twelve-primitive joins: 3, 7, 8, 11). The system's actions are reconstructable in order: reads, writes, tool calls, refusals, memory updates, external outputs.
> 4. **Named owner** (twelve-primitive joins: 1, 2, 6). A human, not a team mailbox, carries the result of the action and the duty to answer for it.
>
> If those four live only in a document, you are still on the PDF.

Run the four checks against Ms Adeyemi's request for customer C-1842 in that one hour, and the test stops being abstract. Per-claim provenance: the line "no refund was issued" must carry a trace ID that resolves to the `refund.issue` call site (or its absence) in the action ledger, not to an analyst's recollection. Authority chain: the line "the agent's draft was held for human review" must show the live refund policy version, the threshold rule it tripped, and the approver who rejected the autonomous issue. Decision trace: the reads of the CRM profile and order O-9912, the policy retrieval, the refund.quote call, the draft email and the working-memory

write must reconstruct in order. Named owner: the Head of Support Ops takes the consequence, not a team mailbox. Any of the four that lives only in a binder is the line of the answer Ms Adeyemi will push back on.

There is a fair objection waiting at the end of that exercise: auditors trained on documentation cannot read runtime evidence. That is true today and worth admitting. The answer is to ship runtime evidence in the format the documentation auditor already knows: per-claim provenance, signed traces, named owners, a trace identifier that resolves to the same artefact in every report. The bridge is the format, not new auditors. A pile of JSON spans will defeat a regulator who came expecting a binder; a binder generated from spans, with each claim hyperlinked to its trace, lets the same regulator do the work they already know how to do, faster.

And because the format carries its own integrity, the append-only storage, the signing at write time and the independent verifier from the three-kinds-of-evidence section, the binder survives the next question after that, the one a plaintiff's lawyer asks: how do we know the system did not write itself a flattering record? The trace the auditor reads is one the agent could not have edited to suit the moment.

Why the regulatory stack will not wait for your diagram

The first mistake organisations make is to go looking for the AI regulation, as if there is one, the way one might go looking for the off switch. There is no single regulation and there is no single off switch. Agents are too general for any one regime to cover them. The same evidence regime is reinforced from every adjacent instrument that touches the agent: cyber-resilience and network-security law on the security side, the data-sharing rules on the data side, the platform-services and product-liability regimes on the liability side [5]. Each carves out its own slice of the logs the agent is already producing. What Nannini and colleagues do crisply is refuse to treat agents as a special category needing brand-new law. They ask instead what existing law already requires, and their conclusion matches the engineering one exactly: the regulatory triggers follow the agent's external actions, its data flows, the systems it connects to, and the people it affects.

Vendors keep trying to win the argument by naming the box differently, "it is not an agent, it is an agentic workflow," "it is not high-risk, it is a copilot." The regulator looks past the name at the actions, the data flows, the connected systems and the affected persons. If those cross a threshold, the regime applies, whatever the box is called. And the frame travels across borders without losing its shape. Whether the questioner reads the system through Europe's AI rules or California's AB 316, through the US risk-management framework's governance function or Canada's fair-information principles [15][16], the demand resolves to the same runtime evidence: the trace the system must produce does not change with the letterhead.

What does change with the letterhead is the calendar. Three dates anchor everything that follows in this section, one per regime, so the table below is worth keeping at hand rather than reading once and forgetting.

EFFECTIVE DATE	JURISDICTION	WHAT CHANGES
1 January 2026	California, AB 316	The autonomous-AI defence is eliminated in covered civil actions.
2 August 2026	EU AI Act enforcement powers	Enforcement architecture and general-purpose-AI governance powers become live: fines, information requests, and model recalls become tools for the European Commission [13]. High-risk application dates are separately in flux (see the deferral discussion below).
DSIT publication 31 Jan 2025; ETSI publication April 2025 (TS 104 223 V1.1.1, announced 23 April 2025)	UK DSIT AI Cyber Security Code of Practice (adopted as ETSI TS 104 223)	The UK and ETSI baseline for AI lifecycle security [14].

Europe's framework anchors that stack. Event-recording, human oversight with a real stop-or-reverse, quality-management, the rules on substantial modification, operator monitoring and log retention, post-market monitoring, incident reporting and explanations all point in the same direction: runtime evidence rather than retrospective prose [1]. The transparency obligations are now being filled in too; on 8 May 2026 the Commission's AI Office opened a consultation on draft guidance for them, running into June [12]. The high-risk calendar, by contrast, is uncertain. The Commission and Council have described a provisional political agreement to simplify the rules, the Digital Omnibus, that would move the application dates to 2 December 2027 for stand-alone high-risk systems and 2 August 2028 for high-risk AI embedded in products, but formal adoption and publication still have to be confirmed before those dates can be treated as binding law [9][10]. A deferral gives you runway, not relief. Whatever the date turns out to be, the regulation is moving towards action-based scrutiny of agentic systems, and the organisations that are not sanctioned will be the ones that built runtime evidence into the system before the date arrived.

California supplies the simpler sentence. AB 316 was chaptered on 13 October 2025 and took effect on 1 January 2026, and it blocks the narrow "the AI autonomously caused the harm" defence while leaving other defences intact [2]. That does not make every bad agent action automatically actionable. What it does is turn the action ledger from a debugging convenience into something heavier: the future record of causation, scope, authority and correction, in a proceeding where the operator, not the agent, is the defendant. "The model decided" is no longer a line anyone can rely on. Vendors describe architecture. Regulators, courts and affected people ask for records.

Memory is where the neat compliance story falls apart

When a person invokes a data right, can you prove you actually let go of them? That is the eleventh primitive, and it exposes the old PDF model more brutally than any other: forgetting.

Picture a customer who has interacted with an agent for six months. Over that time the agent has summarised the customer's preferences, retrieved their past tickets, written running notes about each encounter, stored numerical representations of their data in a vector database, the kind of store that lets a system find records by similarity rather than by exact match, cached retrieved context, and perhaps fed a helper model that learned from those same interactions. Then the customer exercises a data right and asks to be forgotten. What does "forget me" actually mean here? It is plainly not enough to delete the customer's row in the CRM, the customer-records system, if operational memory still holds summaries, vectors, cached prompts, evaluation traces and downstream copies of the same person. The right to erasure has been enforceable law in Europe for years, and it was an awkward fit with statistical models from the start. With agents the awkwardness becomes acute, because the customer's data lives in places nobody planned for.

The regulators have noticed, and on this question their guidance is the legal weight to lean on. France's data-protection authority, the CNIL, treats individual rights in AI systems as an operational problem rather than a slogan [3][6]. The European Data Protection Board's 2025 coordinated enforcement report on the right to erasure widens the lesson: procedures, staff training, exceptions, retention schedules, backups, anonymisation and the processes for proving deletion are all recurring practical failure points, not paperwork footnotes [7]. And the UK regulator, the ICO, makes the point in the same operational register in its guidance on erasure and backups [8]. That supervisory triad, the CNIL, the EDPB and the ICO, is the doctrine to lean on here; a separate research preprint is useful only as vocabulary, naming the memory surfaces plainly through a lifecycle of write, store, retrieve, execute, share, and forget or roll back [11].

This is why the eleventh primitive has two halves, forget and verify. A memory-forgetting record should say what was deleted or revoked, where, when, by whom or by which automated job, which downstream stores were updated, and what verification probe was run afterwards to confirm it. Concretely, for customer C-1842: re-issue the original similarity query against the vector store and confirm that zero records resolve to that customer; grep the cache and the eval-trace stores for the customer key and confirm both come back empty; and re-run the agent on a fixed prompt that previously surfaced the customer's preference and confirm it no longer does. The discipline is that the probe is a test which fails before deletion and passes after; a check that cannot fail before deletion proves nothing and merely reassures. If deletion was delayed, incomplete, or constrained by some other legal obligation, the system should preserve that explanation too.

Memory governance is not "do we have memory?" It is "which memory phase is governed, recorded and reversible?", across write, store, retrieve, execute, share and forget. Forgetting without evidence is only not-looking.

A compliance leader reading this list has a reasonable retort, and it deserves a real hearing. Compliance teams already do governance. They run impact assessments, keep policy binders, negotiate data-processing terms, review audit reports, maintain risk registers and prepare audit packs. The twelve primitives can sound like an engineering team reinventing that machinery in its own language. Part of that objection is right, and the answer is that most of the work in the list should connect to the machinery that already exists rather than duplicate it. The agent registry should feed the risk register. The data-flow inventory should feed privacy operations. The action ledger should feed incident response. The evaluation records should feed quality management. The audit-ready trace store should feed the team that answers regulators. The genuine difference is timing and granularity. Existing compliance artefacts describe intended control. Agentic governance also has to record action-time control: which policy was live, which tool was called, which approval screen a human was shown, which memory was retrieved, which version acted, and which reversal path existed at that exact instant. None of that duplicates bureaucracy; it supplies the missing runtime half of the bureaucracy you already have.

I want to be exact about what the legal evidence does and does not do. It does not bless this precise list of twelve. It does something more useful: it points at the same place from several directions at once. Europe's AI rules create traceability, oversight, quality-management, log-retention, monitoring, incident and explanation duties for high-risk systems [1]. France's authority [3][6], the UK regulator [8] and the European Data Protection Board [7] turn erasure and AI data rights into operational evidence questions rather than slogans. California's AB 316 removes the AI-autonomy defence, but only in the narrow way the statute states [2]. Nannini and colleagues explain why an agent in Europe sits across many regimes at once and why its external actions, data flows, connected systems and affected persons become the compliance object [5]. The Agent Payments Protocol shows in a single domain how mandates and bounded authorisation can become runtime evidence rather than after-the-fact narration [4].

Four questions this argument cannot answer, and is honest about not answering:

1. What minimum trace granularity will European regulators ultimately accept for highly autonomous, tool-using agents?
2. How aggressively will "substantial modification" be read against agent stacks that change continuously as models, prompts and tools are updated?
3. How quickly will harmonised standards and Commission guidance settle the exact shape of the high-risk evidence layer?
4. Is there public evidence of any high-risk AI deployer that has disclosed all twelve of these primitives running in production? I know of none.

That last gap is not a reason to wait. It is the reason to start, because the first organisations to build the evidence layer will define what "enough" looks like, and the rest will be measured against it.

Return, then, to Ms Adeyemi. Suppose her email arrives at the second kind of organisation, the one that built the twelve primitives in rather than bolting a binder on. The team does not forward her note to the vendor and wait. Someone runs a query. By Thursday afternoon there is an answer: for customer C-1842, between two and three o'clock, the agent read these four records, called these three tools, was refused the fourth, routed one decision to a named human who rejected the autonomous refund, wrote one working summary that has since expired, and sent nothing outward. Every line of it carries a trace identifier she can follow. What reaches Ms Adeyemi this time is not a screenshot, not an emoji, but a record, the one thing a cordial regulator and an angry customer both, in the end, actually want. The difference between the two organisations was never the quality of their intentions. It was whether the evidence existed before anyone asked for it.

So take the question into your own building. If the regulator's email arrives next Wednesday, which of the twelve primitives can your system actually produce by Friday, for one named customer, one named hour and one named agent? And which primitive does not yet exist anywhere in your architecture at all?

Carry This Forward

> **One move now.**
>
> Answer those two questions before anyone makes you. Choose one named customer, one named hour, one named agent, and try to produce, from records and not from memory, the data reads, the tool calls, the approvals, the memory writes, the model version, the policy version, the owner and the external outputs. Where a line comes back empty, write down the missing primitive, name the person who owns the repair, and name the verification probe that would prove the repair actually worked, a test that fails before the fix and passes after. Keep that answer close to the system, not buried in a governance binder where it will quietly go stale.

Compliance proves what happened, under whose authority, and with which record. It is backward-looking by nature. The decision a leader faces is forward-looking: given what the system did, does it deserve more authority next time, or less? That is the question of evaluation, and it is where the next essay goes.

Three regulatory lenses US · EU · UK

Operating questions, not legal advice. The frameworks stay the same; the regulator changes.

US Can you prove what ran, what data was touched, what authority existed, what happened, and what remedy exists, under NIST RMF, FTC, state liability and sector rules?

EU Can you prove the same five facts under EU AI Act enforcement powers (effective 2 August 2026), GDPR, the Data Act and product / cyber duties?

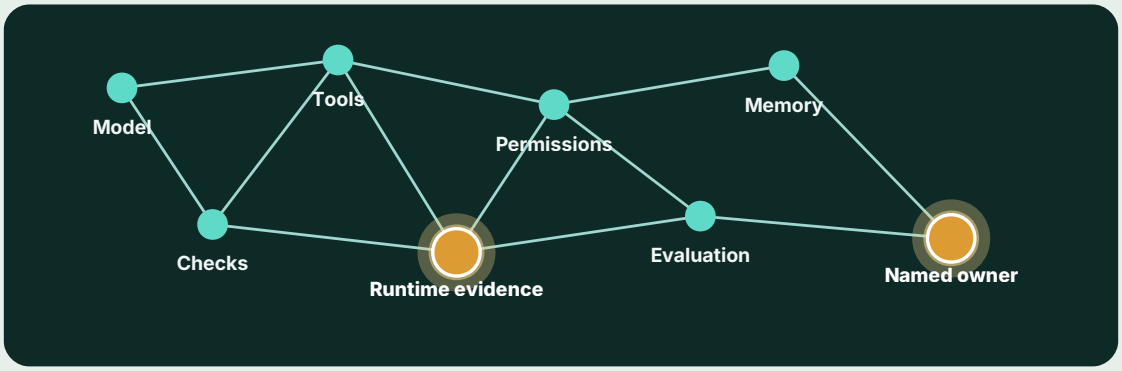
UK Can you prove the same five facts to sector regulators, the ICO and the DSIT AI Cyber Security Code, with the records they expect?

CANADA Canadian deployers should track PIPEDA, provincial regimes (e.g. Quebec Law 25), and the present gap in Canadian federal AI law.

THE STACK SO FAR E19 · Essay 19 of 22 complete · Arc IV: Proof and accountability

The Stack So Far. *Every essay adds one instrument to the operating model. The constellation shows which eight you are building, which are lit by essays you have read, and which is added right here.*

I See the object | II Evidence and authority | III Runtime control | **IV Proof and accountability** ESSAY 5 OF 5 | V Operating model



built in earlier essays | added in this essay | coming in later essays

01 02 03 04 05 | 06 07 08 09 10 | 11 12 13 14 | 15 16 17 18 **19** | 20 21 22

Arc IV complete. You can now reconstruct proof and accountability. Run trace, semantic failure, witnesses, identity, compliance.

You have just added.

The twelve runtime compliance primitives

You can now convert compliance into runtime evidence.

Next. E20 asks how to evaluate an agent for promotion, not as a leaderboard entry.

← PREVIOUS
E18 · When an Agent Acts, Who Acted?

Essay 19 of 22 complete

NEXT →
E20 · You Cannot Benchmark a Coworker

References

Reference links for sources cited in this essay.

1. **EU AI Act, Regulation 2024/1689**
   European Union
   https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

2. **California AB 316**
   California Legislature
   https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=202520260AB316

3. **Ensuring and facilitating data-subject rights**
   CNIL
   https://www.cnil.fr/en/ensuring-and-facilitating-exercise-data-subjects-rights

4. **Agent Payments Protocol specification**
   AP2
   https://ap2-protocol.org/ap2/specification/

5. **AI Agents Under EU Law**
   Nannini et al.
   https://arxiv.org/abs/2604.04604

6. **CNIL AI data-subject rights guidance**
   CNIL
   https://www.cnil.fr/fr/node/165882

7. **EDPB CEF 2025 Right to Erasure report**
   European Data Protection Board
   https://www.edpb.europa.eu/system/files/2026-02/edpb_cef-report_2025_right-to-erasure_en.pdf

8. **Right to erasure: backups**
   ICO
   https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/individual-rights/individual-rights/right-to-erasure/

9. **EU Digital Omnibus agreement**
   European Commission
   https://ec.europa.eu/commission/presscorner/detail/en/ip_26_1024

10. **Council press release on AI simplification**
    Council of the EU
    https://www.consilium.europa.eu/en/press/press-releases/2026/05/07/artificial-intelligence-council-and-parliament-agree-to-simplify-and-streamline-rules/pdf/

11 **A Survey on the Security of Long-Term Memory in LLM Agents: Toward Mnemonic Sovereignty**
Lin et al.
https://arxiv.org/abs/2604.16548

12 **Consultation on draft guidelines on Article 50 transparency obligations of the AI Act**
European Commission AI Office
https://digital-strategy.ec.europa.eu/en/consultations/consultation-draft-guidelines-transparency-obligations-under-ai-act

13 **EU AI Act enforcement timeline**
European Commission
https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

14 **UK DSIT AI Cyber Security Code**
DSIT (UK Government)
https://www.gov.uk/government/publications/ai-cyber-security-code-of-practice

15 **Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1: Govern function**
National Institute of Standards and Technology
https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf

16 **PIPEDA Fair Information Principles (Principles 4.5, 4.8, 4.9)**
Office of the Privacy Commissioner of Canada
https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/

About the Author

ARCHITECTING THE AI COWORKER

Dr Peter McCann Strain

Dr Peter McCann Strain is a CTO, founder, and senior AI engineer with a DPhil/PhD in AI from Oxford University. He builds production AI systems and writes about making agentic AI useful, inspectable, governable, and safe enough for real work.

READ THE FULL SERIES

Substack (canonical)	petermccannstrain.substack.com
Medium	@peter.mccann.strain
LinkedIn	peter-strain-dphil-15a607128
Web	petermccannstrain.com
Cadence	New essays twice weekly, 2 June – 21 July 2026