Contents

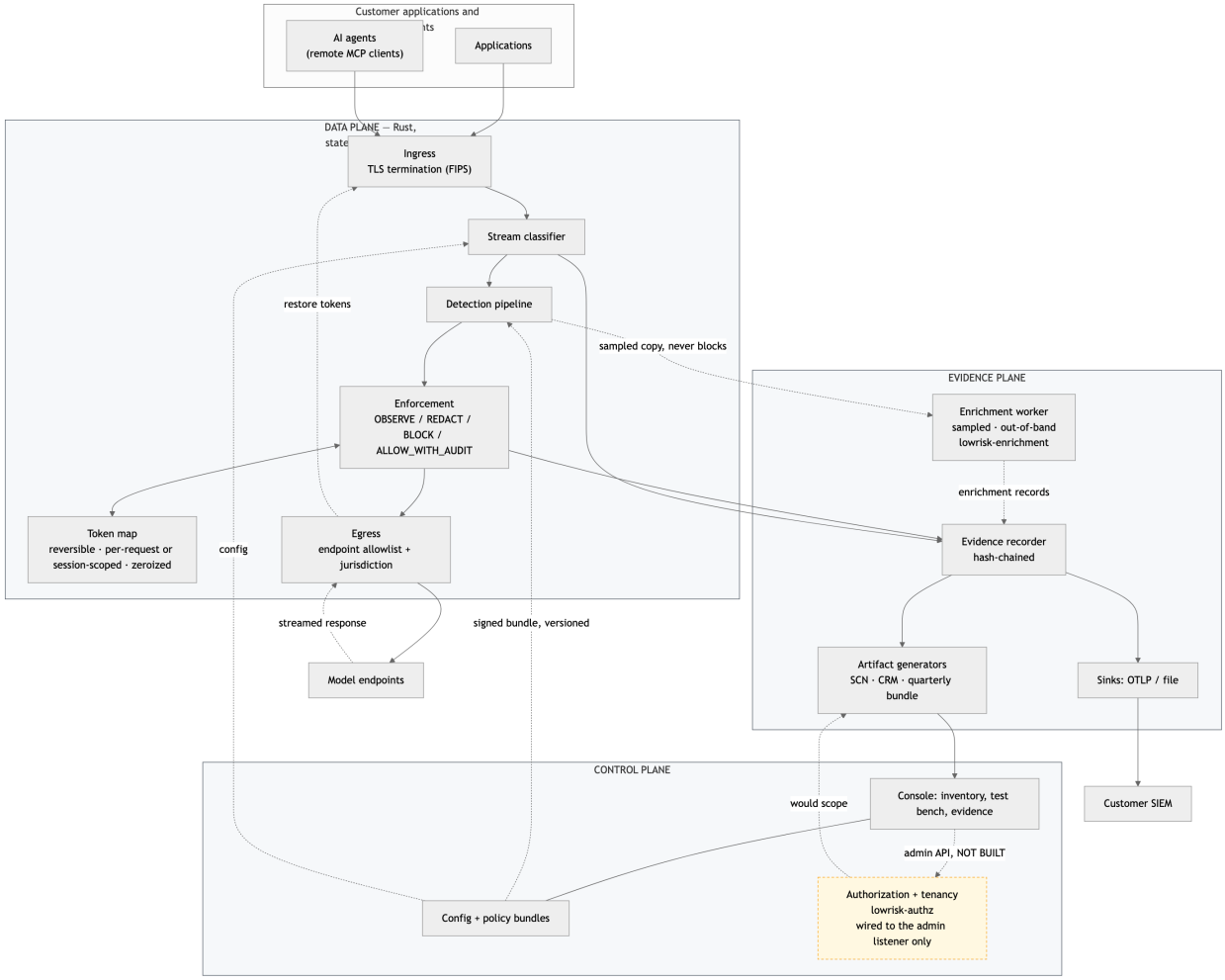
LowRisk.io — Architecture

Applies to: PRD v4.1.0 **Audience:** engineers building it, and assessors evaluating it. Both are first-class readers; where they need different things, the assessor's need is called out in a **Evidence** note.

---

1. Shape of the system

Three planes, deliberately separable. The data plane must keep working when the other two are gone.



Two edges in that diagram are **dashed because nothing traverses them today**, and the distinction is load-bearing rather than cosmetic:

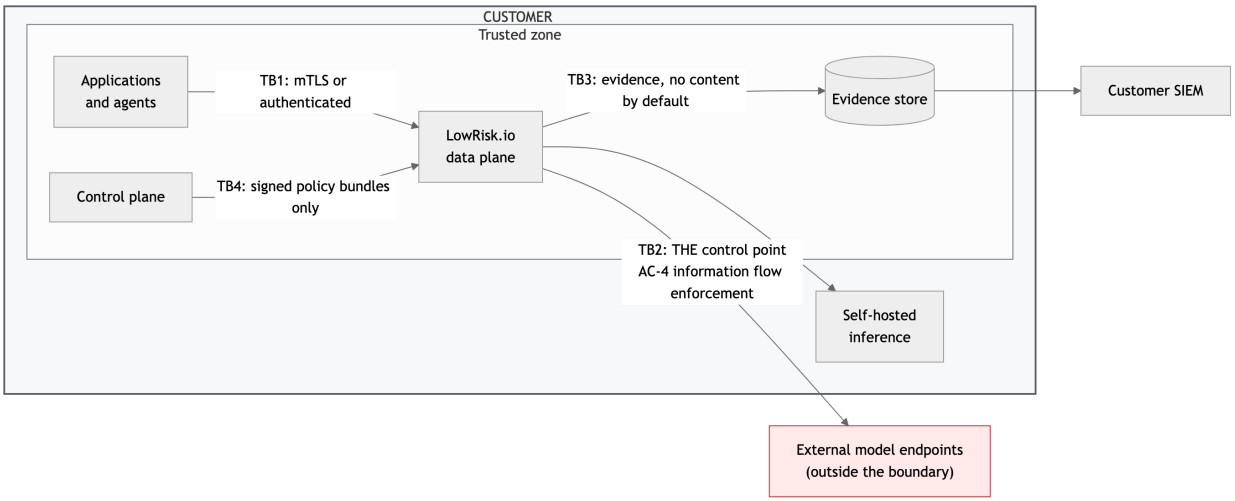
- `UI -.admin API.-> AZ` — still dashed. The console's admin API does not exist. `lowrisk-authz` DOES now have one production caller (the proxy's own admin listener, §2.1), so the crate is no longer inert — but the edge this arrow draws, console to authz, carries nothing. Solidifying it because a *different* caller appeared would be the same defect as a gate that cannot fail (decision 0018), in a picture.
- `DET -.sampled copy.-> ENR` — deliberately out-of-band. Enrichment findings are advisory and carry a `Provenance` marker recording that they could not have affected the decision.

Why the split matters. The control plane going down must never take AI traffic with it — the data plane runs on its last valid configuration and alarms (§7.3 of the PRD). The evidence plane is separated because its failure mode is different: evidence loss is a compliance event, not an availability event, and it is handled by the failure policy rather than by dropping records silently.

---

2. Trust boundaries

The boundary diagram is the one an Authorizing Official will actually look at. Everything crossing a boundary is labelled with what is enforced there.



Boundary	What crosses	Enforced by	Controls
TB1 app → proxy	Prompts, tool calls, credentials	Stream classification and identity resolution	AC-4, IA-3, AC-3
TB2 proxy → external model	**The only path sensitive data leaves the boundary**	Detection, enforcement mode, endpoint allowlist with jurisdiction	**AC-4**, SC-7, SC-7(5), SC-7(8), SI-4(4)
TB3 proxy → evidence store	Evidence records. **Never prompt content unless explicitly opted in**	Telemetry policy, hash chaining	AU-3, AU-9, SC-8(1)
TB4 control plane → data plane	Configuration and policy bundles	Signature verification, version pinning	SI-7, SI-7(1), SR-11

Evidence. TB2 is the claim the whole product rests on. The deployment guide's bypass-detection procedure (REQ-INV-5) is what turns "traffic goes through us" from an assumption into a measured statement — and where it can't be measured, the compensating control is named in the AO one-pager rather than left implicit.

2.1 Admin principal authentication — what each source can honestly attest

Status: both sources are live, and the recency tier is reachable. As of 2026-07-27 the proxy authenticates an administrative principal from a client certificate *or* an OIDC bearer token and authorizes it through `lowrisk-authz`, which until that day had a complete model and **no runtime effect at all**. Decisions 0021, 0023, 0024 and 0025 are the reasoning; REQ-SEC-21, REQ-SEC-22, REQ-OPS-8a and REQ-EVL-2 are the requirements.

What is live: the admin listener has TLS at all (it was handed none, so a deployment with mutual TLS still served `/signals` in plaintext); the certificate policy extension is parsed at handshake; `Authorizer::check` gates `/signals` and `/whoami`.

The two listeners use different client-auth policies, and the difference is load-bearing in opposite directions. The data plane demands a client certificate at the handshake — optional client auth on the path that carries stream identity is not authentication. The admin listener **permits anonymous**, because an orchestrator's probe presents no certificate and cannot be configured to; a mandatory verifier there makes a pod that never becomes ready, which is the same failure this decision fixed in the Helm chart. The application refuses privileged endpoints instead, which it does regardless. A certificate that *is* presented is still verified against the same CA: anonymous permitted, not unverified accepted.

OIDC landed too (REQ-OPS-8a). A second source, and the only one that can emit `AssertedAt`. Its shape was decided by the requirements rather than chosen (decision 0024): keys are read from a **file** and never fetched, because REQ-OPS-3 forbids a runtime network callback and REQ-OPS-4 permits outbound calls only to model endpoints and customer sinks; they reload without a restart (REQ-SEC-6), re-read on file change *and* on an unknown key id; verification uses the crypto module already bound at build time, so authentication stays inside the FIPS boundary; and **no symmetric algorithm is implementable**, which makes the RS256→HS256 confusion class unrepresentable rather than defended against.

The recency tier is live. `POST /evidence/erase` is the first destructive administrative action and the first to demand `FactorRequirement::Recency`, so decision 0021's `MfaRecencyUnknown` refusal is now reachable from a real request: a PIV Authentication certificate is refused there unless the deployment records an acknowledged air-gap waiver. Both branches verified against a running proxy.

What is still not: no group-to-role mapping, SCIM, session policy or access-review export (REQ-OPS-8b–8e); only one action demands recency, because only one destructive action exists; and the rest of `lowrisk-authz` — tenant derivation, cross-tenant checks, access review — still has no caller.

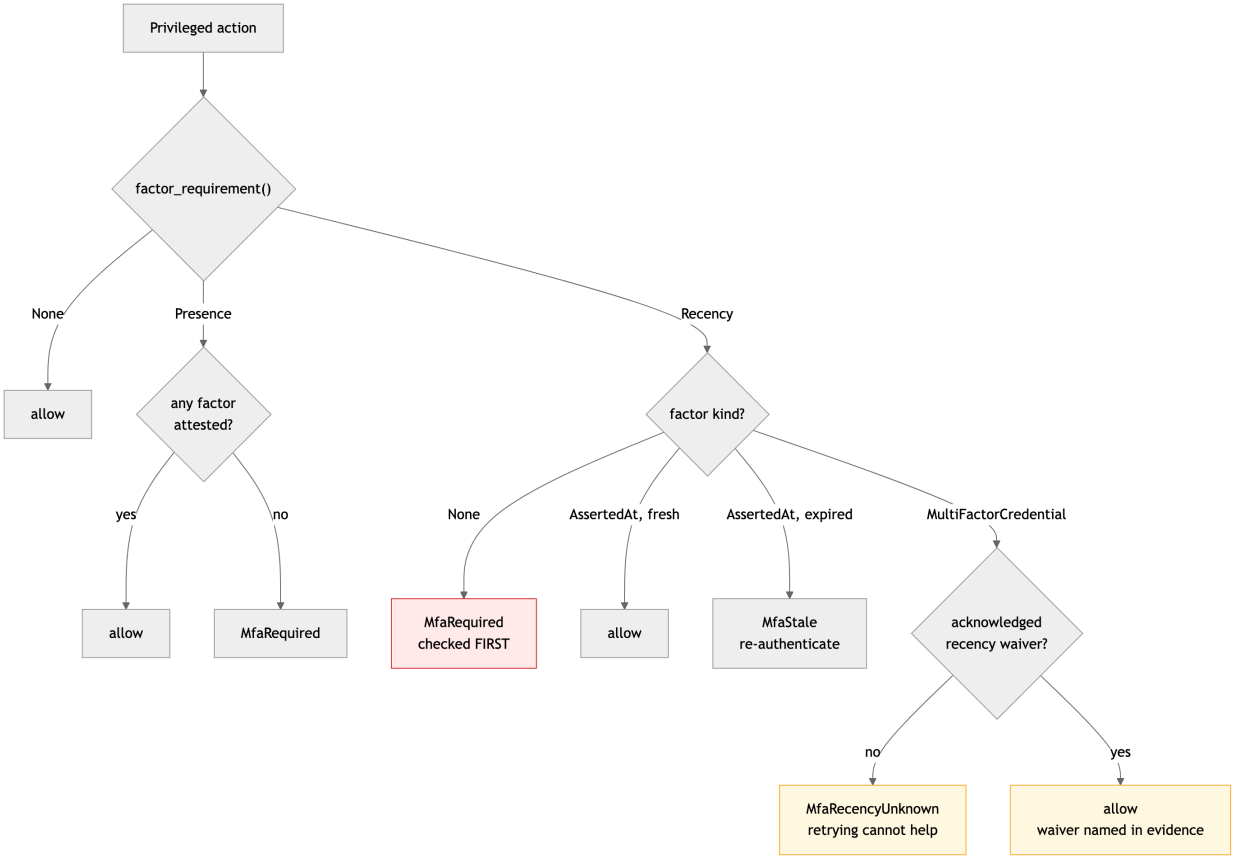
`auth_time` is OPTIONAL unless the client requested `max_age`, so a federated deployment may be no better placed to prove recency than a certificate one. The type already handles that: absence is reported, never inferred.

The product is customer-deployed inside the customer's authorization boundary, so the identity infrastructure is **theirs** and differs per topology:

Topology	Reachable IdP?	Realistic admin auth	Can prove recency?
Air-gapped / classified enclave	**No**	Client certificate. Nothing else works	**No**
On-prem VPS, small agency	Maybe	Client certificate, or local accounts	Only via an IdP
Agency cloud, Entra / Okta / Login.gov	Yes	OIDC	**Yes** — `auth_time`
Contractor-operated for an agency	Yes, theirs	OIDC to *their* tenant	Yes
Future LowRisk.io SaaS	Ours	Sessions with SSO/MFA	Yes

Plurality is therefore a requirement, not a preference. A product mandating OIDC cannot be deployed air-gapped; one mandating mTLS cannot use the agency's existing SSO, and an ISSO will ask why privileged access to the evidence plane sits outside their IdP.

The crux. SP 800-53 IA-2(1) requires MFA for privileged accounts, and PIV is how the government usually delivers it — so mTLS is not a downgrade from federal MFA, it is the mechanism federal MFA arrives on. But **a client certificate cannot tell you when the second factor was asserted.** The PIN was entered when the key was first unlocked, possibly hours ago. The handshake proves possession *now*; nothing in the certificate or the TLS session carries recency.

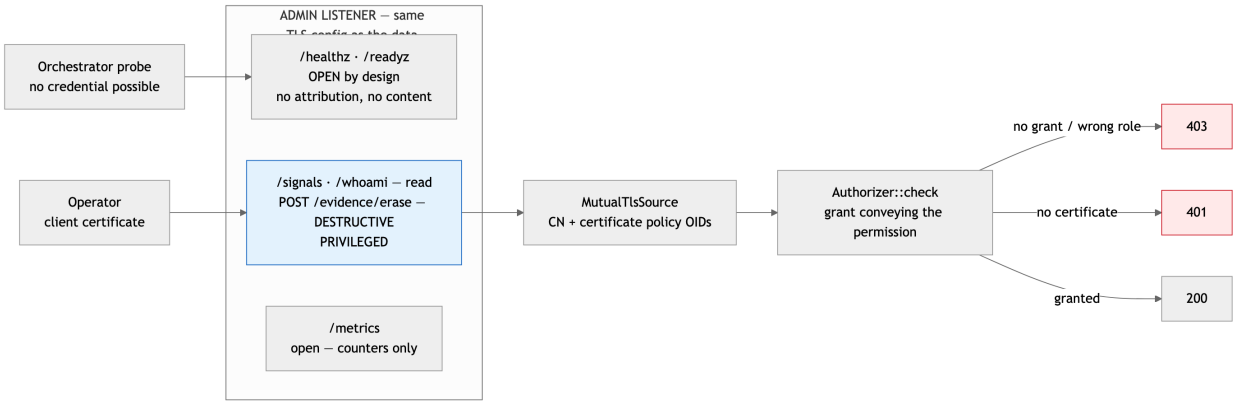


Three properties in that diagram are load-bearing:

- **`FactorEvidence::None` is checked before the waiver**, so a waiver meant to accept PIV can never decay into "MFA off". Proved by `a_waiver_rescues_a_certificate_but_never_an_absent_factor`.
- **`MfaRecencyUnknown` is a distinct denial from `MfaStale`.** Stale means re-authenticate; unknown means the deployment's authentication source structurally cannot answer and retrying will never succeed. Telling an operator to retry something that cannot work is worse than refusing plainly.
- **The waiver requires a name and a reason**, constructed through a fallible constructor so a blank one is not expressible — the same shape as `fail_open_acknowledged_by`.

The alternative — `Option<u64>` with an mTLS backend writing `Some(now)` — makes every freshness check permanently true while the code still *looks* like it enforces step-up. That is this project's recurring defect, and the type is what forecloses it.

The admin surface, in two tiers



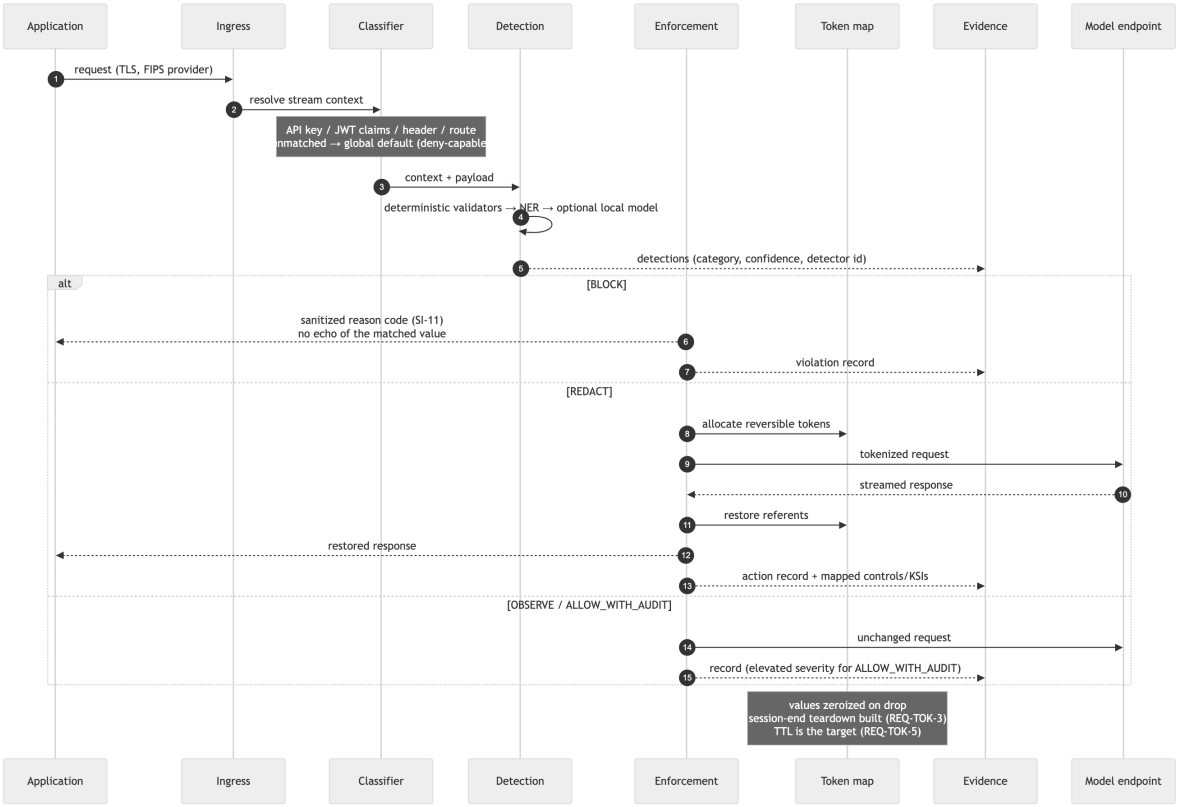
Probes are open **deliberately**. An orchestrator reaches them with no credential and no way to present one; requiring authentication there produces a pod that never becomes ready, and the operator's rational response is to disable the probe — trading a real availability control for a nominal access control. So they are open and carry nothing worth protecting.

This is also where the chart was wrong: both probes pointed at the **data plane**, which answers `/healthz` with `403 destination not permitted`. The pod never became ready and every attempt wrote an evidence record, so a running deployment would have filled the hash chain with its own health checks.

Sequencing: mTLS first — done — because it is the only mechanism that works in every topology, and the proxy already extracted a client-certificate subject in `identity.rs`. OIDC second, and **the `AuthenticationSource` trait is not frozen until both exist**: they differ on precisely the axis that matters, so building both is what forces the abstraction to be honest. The trait exists and has one implementor; that is not the same as being settled.

---

3. Request lifecycle



REDACT is a reversible round-trip today — per-request (decision 0066). The `E->>M`, `E->>T`, and `E-->>A` steps are live: each detected span is replaced before forwarding with an opaque per-request token `{{LR:CATEGORY:N}}` — the same value mapping to the same token within the request, so the model sees a consistent referent — the token map is retained for that one request, and the model's response is detokenized on the way back so the caller receives the original value and the provider never held it. Only categories a REDACT rule matched are tokenized — a value left on OBSERVE is recorded, not modified. This replaced the one-way `[REDACTED]` marker of decision 0062: there is no longer a category-blind irreversible mode, and the token now discloses the value's category to the provider by design (the reversible use case requires it; the category-free marker did not).

The map is now optionally SESSION-scoped (decision 0088, REQ-TOK-1). A stream that opts in (`token_map.session_scoped`) keeps its token map for the whole stream **session**, named by an explicit `x-lowrisk-session` header, so a value carries the same token across the session's requests and the model reasons over one referent across turns rather than only within a request. A `SessionRegistry` in the proxy owns one map per session id; a different id gets a different map and no path hands one session's map to another (the **SC-4** ownership invariant, REQ-TOK-2's seam), and a blank id is normalized to absent (falling back to per-request) after an adversarial review found an empty header would otherwise pool distinct callers into one shared map. The registry is **bounded** — session ids are caller-supplied, so past a fixed cap a new session degrades to per-request (surfaced by a metric) rather than growing memory unboundedly, the decision-0026 hazard. A stream that does not opt in is byte-for-byte the 0066 per-request behaviour. **Session termination is now built (decision 0090, REQ-TOK-3):** `SessionRegistry::end`, wired to the `x-lowrisk-session-end` request header, removes a session's map so its `Zeroizing` plaintext is overwritten when the last handle drops — a response still streaming for the session keeps its own handle, so an in-flight exchange is never wiped mid-restore, and a reused id after teardown is a fresh session. The buffer-clear is a memory-assertion witnessed on a live allocation (observing a freed buffer would be UB). The remaining REQ-TOK increments stay planned: REQ-TOK-4 (an affirmative disk / telemetry exclusion statement —

the map has no `Serialize` and its `Debug` prints only counts today), and REQ-TOK-5 (per-stream TTL, the "timeout" trigger, which would let the live-session count fall by AGE rather than only on an explicit end). Since decision 0089 the SC-4 ownership invariant (REQ-TOK-2) is proven by a CONCURRENCY test: on a multi-thread runtime with more session tasks than workers, each rescheduled mid-conversation, a session handed a token only another session issued must pass it through unchanged — the enforcement mechanism is the serializing proxy mutex, so the test exercises task migration and interleaving rather than parallel mutation, which the mutex forecloses. Response-side DETECTION is likewise unbuilt: a secret the model GENERATES in its own response — as opposed to a token it echoes, which IS restored — is recorded, not removed mid-stream, because bytes already delivered cannot be recalled.

Streaming. Responses are inspected incrementally. Detection must never buffer a full response to make a decision — it would destroy time-to-first-token, which is the latency users actually perceive. This constrains the detector interface to operate on a sliding window with carry-over state, and that constraint is load-bearing: it is why detection is a streaming trait rather than a `fn(String) -> Vec<Detection>`.

Forwarding preserves the request — and inspects every channel that opens (decision 0078). The `E->>M` step no longer reshapes the request to a fixed `POST /`: `forward_upstream` preserves the client's method, path, query, and safe headers, so an arbitrary REST or MCP call reaches the allowlisted upstream as itself (`Authorization` carried; `CONNECT` refused — the proxy is not a tunnel, non-goal N2). A **strip-set** makes that safe — `host` is proxy-owned (set to the upstream), `content-length`/`transfer-encoding` are re-derived from the actually-forwarded body (a stale length smuggles), the hop-by-hop set and every `Connection`-named header are dropped, and any `x-lowrisk-` control header is stripped by prefix. The load-bearing consequence: the preserved method, URL, and headers are **egress channels the body inspection never saw** — before 0078 the body was the only exit. So the request line (method + target, inspected raw *and* percent-decoded the way the upstream reads it) and every forwarded header are handed to the **same detector and policy as the body**. Because a URL, header, or method cannot be tokenized and restored on the response the way a body span can (there is no response echo), content the stream would REDACT or BLOCK there is **BLOCKED — fail-closed**; an OBSERVE/AUDIT finding forwards but is still recorded, so a value that leaves on a non-body channel is never invisible to the evidence chain.

The MCP tool gate — a third pass on designated streams (decision 0080). On a stream the operator designates MCP, a third inspection runs between the metadata gate and body inspection, governing a fact the content detectors cannot express: *which tool* a JSON-RPC `tools/call` invokes. A call naming a tool outside the stream's allowlist is denied with a sanitized reason and an evidence record. The gate forwards the **raw** body and decides on the **parsed** view, so its residual risk is a parser differential — an upstream that resolves the same bytes to a different tool. It closes that risk by refusing *ambiguity* rather than only disallowed tools: an unparseable body, a duplicate object key anywhere, a body past the inspection cap, a non-object where a request belongs, or a batch any element of which is disallowed or malformed all fail closed (an empty body — an MCP SSE-open GET or teardown DELETE — carries no invocation and is permitted). It composes with the body pass (an approved tool's arguments are still content-inspected) and is contained behind the stream's `mcp` config, so general REST never parses a body as JSON-RPC. This is REQ-MCP-2.

The gate gains a URL sub-check on `url_egress` streams (decision 0081, REQ-MCP-4). From the same single parse, every string leaf of the tool-call arguments is scanned for an `http(s)` URL whose host is a blocked egress target — because an MCP tool (`fetch`, `browse`) handed a URL makes the *tool server* fetch it, and an agent can point that at `169.254.169.254` (cloud metadata / SSRF), a private range, or `localhost`. The classifier's invariant is the same one the whole SSRF surface demands: **normalize the authority the way a permissive client reads it, then fail closed on anything not confidently public** — so alternate numeric encodings (decimal/octal/hex) are denied wholesale, IPv4-mapped IPv6 is folded to the v4 rules, and an un-normalizable host is denied rather than allowed. A blocked URL yields a sanitized `LR-2004` denial that never echoes the URL. The residuals are structural and named: a tool server FOLLOWING a redirect to a private range is beyond a request-side proxy's view, and DNS rebinding / NAT64 need resolution or infrastructure the proxy defers. **Decision 0084 extends the same pass to the SCHEME:** the classifier now returns a *verdict* (`BlockedRange` vs `BlockedScheme`) rather than a bool, and a tool URL using a non-web scheme it must not dereference — `file:` (local-file read / LFI), `gopher:`/`dict:` (protocol-smuggling SSRF

pivots), `ftp:/ldap:/tftp:/smb:` (internal services) — is refused with a distinct `LR-2007`. The scheme set is a DENYLIST, not an allowlist, on purpose: an allowlist would over-block ordinary path arguments (a Windows `C:\Users\…` parses as a one-letter scheme), which MCP tools routinely pass; the cost is that an unlisted scheme passes (a named residual). A denylisted scheme is refused unless its remainder begins with a space (the one natural-language `"file: report"` form), which — after an adversarial finding — is what catches `file:%2fetc%2fpasswd` and `file:etc/passwd` that Python `urllib` decodes to `/etc/passwd`.

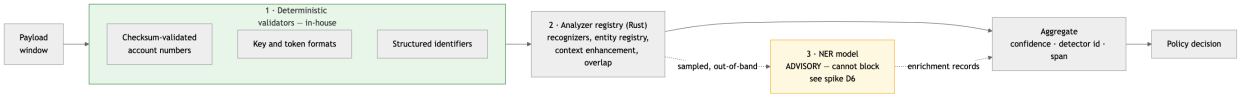
The gate gains a parameter sub-check on `param_rules` streams (decision 0082, REQ-MCP-3 / SI-10). From the same single parse, an allowlisted `tools/call` has the *structure* of its `params.arguments` validated against per-tool rules — because an approved tool can still be handed arguments it was never meant to receive (`search{query, callback_url: "http://attacker/collect"}`: an approved tool, a public host, no sensitive value — the anomaly is the argument *shape*, which no content detector expresses). Three constraints: a parameter allowlist (deny-unknown), required parameters, and a per-value length bound (a string by its bytes, any other value by its serialized length). It is deny-by-default and fail-closed — absent-arguments-with-a-required-param and a non-object `arguments` are refused, and a casing differential on a parameter name can only false-DENY, never false-allow. A violation yields a sanitized `LR-2005` denial that never echoes the parameter. It composes (a card in a permitted value is still content-BLOCKED) and is per-tool opt-in (a ruleless tool is ungoverned even on the same stream); the loader refuses a rule for an un-allowlisted tool so it cannot be silently dead. The named residual with teeth is that validation is top-level only — the interior of an object-typed parameter is not schema-validated.

The gate gains a token-audience sub-check on `token_audience` streams (decision 0083, REQ-MCP-5 / SC-23). Read as the *first* check — from the forwarded metadata (the `Authorization` header 0078 already carries), before the body is parsed — because it governs the credential, not the body, and applies to every request including the empty-bodied SSE-open GET. A bearer token whose JWT `aud` claim matches none of the stream's expected audiences is refused with a sanitized `LR-2006` — the *token-passthrough* defence: a token minted for a DIFFERENT resource, presented here by a confused client, denied before a server can honour it. The load-bearing caveat, stated in the control statement and manual: the signature is **NOT verified** (the proxy holds no issuer keys), so this catches naive / confused-deputy passthrough of a *genuine* mis-audienced token, not a forger — "correctness remains the server's responsibility." Because the check reads the same `Authorization` headers the upstream receives, it refuses the smuggle vector directly: more than one `Authorization` header fails closed (RFC 7230 forbids it), and the scheme is split on any whitespace so a `Bearer`-tab-token cannot slip past. Opaque non-JWT tokens fail closed; a request with no token is not governed (the server's authn concern). Signature verification (JWKS) and REQ-MCP-6 telemetry remain deferred.

---

4. Detection pipeline

Three layers, per decision D3. Ordered cheapest and most precise first, so most traffic never reaches the expensive layer.



Layer	Precision	Owned	Why
Deterministic validators	Highest — a checksum either passes or doesn't	**Built in-house**	Where competitors relying on pattern matching are weakest, and where a few weeks of work produces a durable advantage. Fourteen deterministic detectors live here across five categories — payment card (Luhn), SSN/ITIN (range), DEA (health checksum), IBAN (financial checksum), the credential class (AWS keys, DB URIs, GitHub/Slack/Google/Stripe tokens, PEM markers, JWTs), and CUI markings (the `CUI//` banner and `(CUI)` portion-mark syntax, decision 0064 — a marking detector, not a CUI determination); four more run in layer 2. NPI and ABA routing moved from this set to layer 2 (decision 0063): each has only a single mod-10-class checksum that ~1 in 10 arbitrary runs passes, too weak to fire CERTAIN on a bare number, so they now fire sub-certain with a nearby keyword. The full entity map is `compliance/detection-coverage.yaml`

Layer	Precision	Owned	Why
Analyzer registry	n/a — orchestration	**Built in Rust**	Recognizer trait, entity registry, context enhancement, overlap resolution. Native rather than adopted because the mainstream framework is Python and cannot live in a static musl scratch image — and because the interface must be **streaming**, which a whole-text API is not (D6)
NER model	Bounded — see PRD §6	Adopted, pluggable, **out-of-band**	Commodity. Zero-shot capability matters because customer-specific entity types get added by declaration rather than retraining. **Spike D6 (2026-07-25) measured 96.6 GFLOP per 4 KB window — 276 TFLOP/s would be needed to fit the inline budget, which an A100 with tensor cores still misses by 2x. Layer 3 therefore runs sampled and out-of-band, and is advisory: it cannot block.** Evidence records carry `Provenance`, inside the hash chain

Evidence. Every detection carries its detector id and confidence into the evidence record (REQ-DET-4), so per-layer attribution survives into the artifact. When an assessor asks "how do you know this was caught by a validator and not a guess," the record answers.

Coverage is a register, not a claim. Which sensitive-data entities the deterministic layer recognizes — and which are deliberately deferred to the contextual (layer 2) or model (layer 3) layers, or skipped — is enumerated in `compliance/detection-coverage.yaml`. The taxonomy is seeded from the DLP consensus (Microsoft Purview SIT, Presidio); the layer-1 bar is ours (a checksum, hard range, exact marker, or a prefix+charset+length tight enough to stand alone). It is a forcing function: `scripts/gen_detection_coverage.py` reconciles the register against the detector ids the binary actually registers and refuses an `implemented` row with no detector behind it — the shape that had REQ-DET-2 marked implemented for a year with no credential detector under it.

Layer 2 is contextual and sub-certain — and rides the confidence hook that al-

ready existed. The analyzer-registry row above is no longer aspirational: `context.rs` supplies the two primitives (proximity-keyword search and Shannon entropy), and four detectors use them — a bare SSN recovered by a nearby "SSN" keyword, an AWS *secret* key recovered by entropy plus an assignment keyword, and — since decision 0063 — a bare NPI or ABA routing number recovered by a nearby category keyword, each moved out of layer 1 because its lone mod-10-class checksum passes ~1 in 10 arbitrary runs and cannot carry CERTAIN on its own. A layer-2 detection is emitted below certainty (0.75 for one signal, 0.9 for two), and because `lowrisk_policy::Policy::evaluate` already selects a rule by `confidence.value() >= min_confidence`, an operator's rule threshold decides whether a contextual finding enforces — no policy change was needed. Both layers run in the same synchronous request path; overlap resolution keeps the more confident finding, so a layer-1 CERTAIN result always wins over a layer-2 one on the same span.

Accuracy is measured, not asserted. The shipped detector set is defined once in `lowrisk_detect::default_registry()` — the proxy, the accuracy harness, and the coverage gate all reason about that one list. The harness (`crates/lowrisk-proxy/src/bin/accuracy.rs`) runs it over a labelled corpus and reports precision and recall per category, per layer, and per detector into `compliance/generated/detection-accuracy-report.json`, naming the corpus and the scoring so no figure is a bare percentage (REQ-ACC-1). It is customer-runnable — `make accuracy CORPUS=<path>` scores your own labelled data to measure recall on your population (REQ-ACC-2). The committed corpus is synthetic (one positive and one hard negative per detector), so the committed report measures precision on hard negatives and guards against regression; real-population recall is unmeasured until the harness is run on real data, and the report says so. A self-test in `make check` proves the harness registers a planted miss, so it can never report a vacuous 100%.

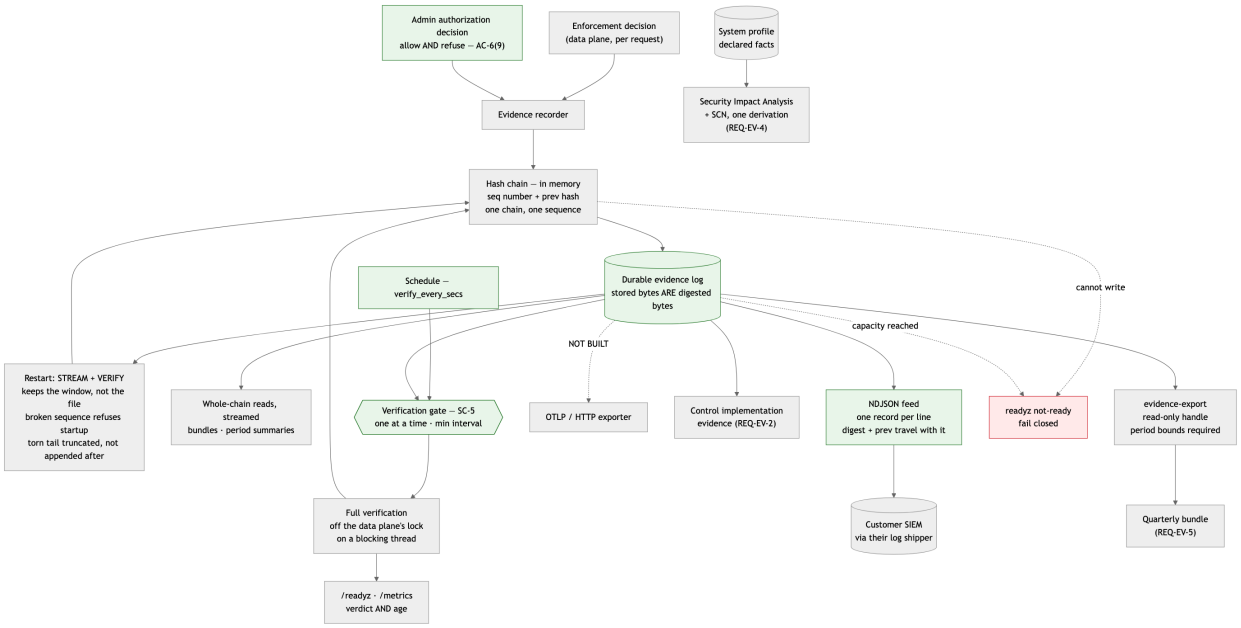
The core never knew it was watching an LLM — the protocol is an adapter (decision 0075). Detection scans raw body bytes, the policy engine decides on the resulting detections, the egress allowlist matches host+port, and the evidence/retention/hold machinery records tamper-evident events — none of it parses `messages`, `choices`, or any LLM-shaped JSON. So the content core is *protocol-agnostic*, and only two things now assume an LLM: SSE response reassembly (`streaming.rs`) and the reversible completion tokenization (decision 0066). The **adapter seam itself is un-hard-coded (decision 0078):** `server.rs::forward_upstream` no longer builds a fixed `POST /` request discarding the client's method/path/headers — it preserves them, so general-REST egress DLP is content inspection the core already does, now actually reaching arbitrary REST. That preservation opened three new egress channels (URL, method, headers) which the same detector/policy now inspects fail-closed (see §3). What remains for MCP-over-HTTP is the rest of the per-tool policy (REQ-MCP-1, 6; the tool allowlist, URL egress, parameter validation, and token-audience inspection of REQ-MCP-2/4/3/5 are built, and REQ-MCP-7's coverage-boundary disclosure is built — decision 0086); it reuses TLS termination, SSE, body inspection, the allowlist, the evidence plane, and now the general-REST forward path unchanged. 0078 is increment (a) — it removes the coupling and completes no REQ-MCP requirement; the MCP adapter is the next slice. The boundaries do not move: this is still first-party *egress* DLP, not browser interception of consumer AI sites (non-goal N2), and MCP coverage is remote HTTP only — a limit the control plane and the generated evidence now STATE affirmatively where MCP inspection is enabled, not merely enforce at load (REQ-MCP-7, decision 0086). See `docs/design/deployment-and-discovery-modes.md`.

---

5. Evidence pipeline

This is the differentiator, so it gets the most design attention.

Read the diagram's line styles. Solid is built and reachable at runtime. Dashed is designed and **absent from the binary**. An earlier version of this diagram drew the buffer, the exporters and the store as though they existed, and the `au-4` control statement was written from that picture rather than from the code (decision 0026). A diagram that does not distinguish the two is how a design becomes a claim.



Integrity model. Each record carries a monotonic sequence number and the hash of its predecessor. Deleting or altering a record breaks the chain at a detectable point; a missing sequence number is a detectable gap. On recovery from an evidence outage, the gap itself is recorded as an event (REQ-FAIL-2) — an absence that documents itself. This is what KSI-MLA-OSM means by *tamper-resistant*, and it is the difference between a log and evidence.

Durability model. Records are appended to a durable log and survive the process (REQ-EVL-15, decision 0028). Several properties are worth knowing before reading anything else here:

- **The stored bytes are the digested bytes.** A record's on-disk form is `Record::canonical()` verbatim followed by its digest. There is no second encoding, so there is nothing for a serializer to get subtly wrong — a file that hashes correctly is exactly what was written, and one that does not fails to load rather than loading modified.
- **Reload verifies before it adopts.** Per-record digests prove nothing was *edited*; only the sequence reveals records removed from the *middle*. A log that does not verify as a chain refuses startup, unless `evidence_log.damaged_acknowledged_by` names somebody who accepted the damage — and that name goes into the chain.
- **Erasure rewrites the file.** Content cleared in memory while the original bytes sit on the volume has not been erased. The rewrite is atomic (temp file, `rename`), so a crash leaves either the old log or the new one.
- **The durability window is a number.** At `fsync_every: 64` a power loss can discard the last 63 records. A clean shutdown flushes regardless, so a rolling deploy loses nothing.
- **The timestamp comes from the chain's declared clock, not the caller — AU-8, REQ-EVL-1 (decision 0071).** Each record is stamped through `Chain::append_now`, which reads the deployment's `TimeSource`; the durable log is attached with the clock `evidence_log.clock` declares, in place of the reproducible test clock a fresh chain starts on. A clock declared `disciplined: true` asserts its timestamps are evidence of *when*, so it loads only under a named acknowledgement and a stated skew bound (decision 0070's idiom); the default is an honest undisciplined wall clock whose skew bound is wide enough that a reader does not trust an ordering it cannot support. Discipline is attested by the operator, not measured in-product — the remaining work on this control.
- **An authorized disposal of the oldest records is a verifiable truncation — AU-11, AU-9, REQ-EVL-19 (decision 0072).** Shrinking a log by removing its oldest records leaves the first

survivor pointing at a predecessor no longer in the file — a non-genesis head the durable verifier reports as a broken link, indistinguishable from tampering or an accidental head-loss. So a disposal writes a *truncation anchor*: a genesis record (`Chain::truncate_prefix`) that stands in for the disposed prefix and declares the digest the survivors point at, carrying who authorized it and through which sequence. Three things make it honest rather than a silenced alarm, and none depend on anyone auditing it: the survivors' bytes are untouched (the anchor bridges the gap by declaration, so a disposal costs one record, not a re-hash of the surviving chain); an *unexplained* truncation that wrote no anchor still reads as a broken link (re-anchoring the survivors onto a fresh genesis would instead make a chopped chain verify as pristine); and the anchor is *attributed*, and since decision 0087 (REQ-EVL-4) *signable*: with an `evidence_log.signing_key` set, the anchor — like every record — carries an ECDSA P-256 signature over its digest, so a forger who rewrites it and recomputes the chain forward is caught, where the unkeyed chain could not. The honest boundary is that the in-process key defends against an external tamperer, not the log writer itself (an HSM/KMS key is the deferred increment). This is the mechanism age-based disposal (REQ-RET-3) needs; the age decision that drives it is built (REQ-RET-5, decision 0073), the legal hold that gates it is built (REQ-RET-4, below), the automated trigger is built (REQ-RET-3, decision 0076), and since decision 0077 the disposal streams a **windowed** file too (bounded memory, refusing only a windowed log that has ever held a legal hold).

- **A legal hold suspends disposal for a named scope, and ambiguity keeps the record — AU-11, AU-9, FRCP 37(e), REQ-RET-4 (decision 0074).** A hold is a chained event with a lifetime, not a flag: `place_hold`/`release_hold` append `HOLD` records carrying issuer, matter, scope, and custodians into the same chain as the evidence they protect, so editing a placed hold to released after the fact breaks the chain like any other tamper. `dispose_expired` (REQ-RET-5) reads the live set (`active_holds`, replayed from the chain, never a cached bit) and stops at the first *held* record — one a scope covers, or a hold's own PLACE record — before it stops at the first not-yet-expired one, disposing only the strictly-expired, strictly-unheld leading prefix. The scope is one of all / tenant / stream / date-range / subject, and the date-range case *widens* a record's timestamp by its clock skew on both sides so a record that merely *could* fall in the held window is kept — the same conservative asymmetry the ceiling boundary uses, pointed toward preservation, because over-retention is not sanctionable and loss under a preservation duty is. A placement with no issuer or no matter is refused, and a release must name a real active hold rather than be silently swallowed. Erasure reaches a hold's personal fields (issuer, custodians) but never its kind, scope, or matter — clearing those would lift a live hold as a side effect of a privacy request. Like the anchor, the hold is *attributed, not signed*.
- **A torn write is truncated, not appended after.** A crash mid-write leaves a partial record. Recovery removes it before opening for append, because a record written *after* an undecodable fragment is a record no decoder will ever reach — either it vanishes into "trailing bytes" on the next load, leaving a shorter chain that verifies perfectly, or the whole log fails to open (decision 0032). The count of discarded bytes is reported at startup; a torn record was never part of any chain.

Memory: bounded at the peak as well as the steady state. `evidence_log.memory_window` bounds the heap — the file holds everything, memory holds the tail — and reads that need the whole chain stream it back from the log, so bundles and period summaries work over a windowed deployment (decision 0031). Startup streams too: the log is fed into the chain a record at a time and only the window is retained (decision 0033). It used to be read in full first, so a windowed deployment restarting on a 6 GiB log allocated 6 GiB at the one moment the log is largest — a restart crash-loop rather than a slow leak. Peak allocation during a load is measured by a test with a counting allocator, not asserted in prose.

Retention and disposition — three tiers, partly built (decisions 0067, 0072, 0073, 0074, 0076, 0077). Retention is configured *per data class*, because the compliance regimes pull the three classes in opposite directions and one period cannot satisfy all of them: **raw session content** (captured only under REQ-TEL-4) minimises toward a near-zero floor; **redacted/tokenised content** sits in a bounded middle; and the **evidence/audit metadata** — the content-free hash chain — carries the long, regime-mandated ceilings (HIPAA/SEC 6 years, SOX 7 years, or the federal cyber-log baseline of OMB M-26-14). Each tier has an admin-configurable floor and ceiling (REQ-RET-1, REQ-RET-2); disposition at the ceiling is real erasure recorded as a tombstone — the same atomic rewrite as above (REQ-RET-3); and a **legal hold** overrides

both floor and ceiling for a named scope, fails safe to preserve, and is itself a chained event (REQ-RET-4). The periods derive from `docs/retention-analysis.md`; the controlling schedule is the deploying agency's, never hard-coded. Disposing raw content early while the content-free chain lives for years is how minimise (GDPR/CCPA) and retain (AU-11) are reconciled. The erasure and durability mechanics above are **built**; since decision 0072 so is the **truncation anchor** (REQ-EVL-19) that lets an authorized disposal of the oldest records verify instead of reading as tampering; since decision 0073 so is the **age DECISION** that drives it — `Chain::dispose_expired` (REQ-RET-5) disposes the oldest records *certainly* past a ceiling, refusing to decide by an undisciplined clock (REQ-EVL-1) and keeping any record within the clock's skew bound of the boundary, because the delete is irreversible; since decision 0074 so is the **legal hold** (REQ-RET-4) that gates it — `place_hold`/`release_hold` are chained events, `dispose_expired` skips any record a live hold covers, and any doubt (a record within skew of a date-range edge) is resolved toward keeping; and since decision 0076 so is the **automated trigger** (REQ-RET-3) that drives the whole arc — an `evidence_log.retention` config block and a background schedule that calls `dispose_expired` on the audit-metadata tier, off by default because a purge is irreversible, and refusing at load every unsafe way to run (no acknowledgement, a zero ceiling, a zero interval, an undisciplined clock). And since decision 0077 the disposal handles a **windowed** log by STREAMING the file — the boundary scan and the rewrite both stream in bounded memory, so windowing is not defeated on the one operation that rewrites the largest file the deployment owns — refusing only a windowed log that has ever written a legal hold (a hold's PLACE record may be evicted from the window and thus invisible, so the header-format gate proves there is none before disposing). The trigger waited for the hold — an automated purge with no hold-suspension is a spoliation exposure — so the audit-metadata tier's enforced disposition is now complete: config, decision, hold gate, schedule, and windowed logs. What is **not yet** built: the per-tier configuration for raw and redacted content (REQ-RET-1/2, awaiting the data itself, REQ-TEL-4), windowed disposal of a log that HAS held a hold (needs a full-file hold replay), and NIST SP 800-88 media sanitization of the underlying device — the trigger reaches the evidence *file*, not the operator's storage medium, backups, or a downstream SIEM copy. Disposing raw content early while the content-free chain lives for years is how minimise (GDPR/CCPA) and retain (AU-11) are reconciled, and an individual-session review path (REQ-REV-1) reads the retained chain for one session where REQ-EV-10 reads the population.

Evidence leaves the node — AU-6. A file sink writes NDJSON for a log shipper to tail (decision 0037): one record per line, carrying the record's digest and its predecessor's so a consumer can reconcile what it received against the chain. Delivery is at least once and the cursor is read back from the sink's own last line, so a crash re-sends rather than skips. An export failure is **degraded, not not-ready** — the log is durable, so an unwritable sink delays forwarding rather than losing a record. The feed carries administrative principals and is as sensitive as the log itself; `/metrics`, by contrast, carries no identity at all.

A verification is a record — AU-9. Every completed full verification appends a `VERIFY` record carrying what it covered, what it found and what started it, with the whole block inside the digest so a result cannot be edited without breaking the chain (decision 0036). It runs on a schedule (`verify_every_secs`, default daily) and on demand, through one code path, so the two cannot disagree. A verification never covers itself — each record attests the prefix before it. The record format was extended without re-anchoring existing chains: the block is appended after the last existing field, so records written before it keep their exact bytes and digests, and the file's format version rises only when a record of the new kind is first written.

The one operation whose cost is the whole log is bounded — SC-5. `POST /evidence/verify` reads every byte and re-derives every digest, and `evidence:verify` is held by six of the seven roles. The permission stays broad: an assessor who cannot ask whether the evidence is intact cannot do the job. So the bound is on the operation — one verification at a time, a minimum interval between them, the read off the mutex the data plane needs and on a thread that is allowed to block. Before decision 0032 that read held the data plane's lock, so one authorised call stalled every proxied request for its duration; a rate limit, which is what was planned, would have made that rarer rather than shorter.

Durability is off by default. Omitting `evidence_log` runs in memory only, which was the sole behaviour before decision 0028. The startup banner states which posture is in force, in capitals when records will be lost.

Administrative actions are evidence too — AC-6(9)

Every authorization decision on a privileged admin endpoint is appended to the **same** chain, as an `ADMIN` record:

Field	Source	Why
`principal`	verified certificate subject or OIDC `sub`	the "who". Capped at 256 chars, truncation marked and disambiguated by digest
`source`	`mtls` | `oidc`	a certificate and an `auth_time` claim support different assertions
`factor`	`FactorEvidence::kind`	`multi-factor-credential`, `asserted-at`, or `none` — "MFA: yes" is not an audit trail
`permission`	`Permission::as_str`	the privilege exercised or attempted
`outcome`	`Permitted` | `Refused { reason }`	**refusals are recorded**; the reason is the authorizer's own code, never a formatted message

Three properties are structural rather than conventional:

- **The same chain, not a second one.** Two chains are two integrity proofs with no established ordering between them, so "who read the evidence, relative to what the evidence says" would depend on interleaving timestamps this system already declares are not reliably orderable (REQ-EVL-1).
- **You cannot authorize without recording.** `AdminDecision` holds the principal and the refusal response in private fields; `record()` appends first and is the only way to obtain either. A handler that skips the audit does not compile. This caught a real defect during implementation — `/whoami` authorized twice, which would have been two records for one request.
- **An administrative record has nowhere to put what the caller saw.** `AdminAction` has no free-text member and no result field, so recording administrative *reads* cannot become a second copy of the data being read.

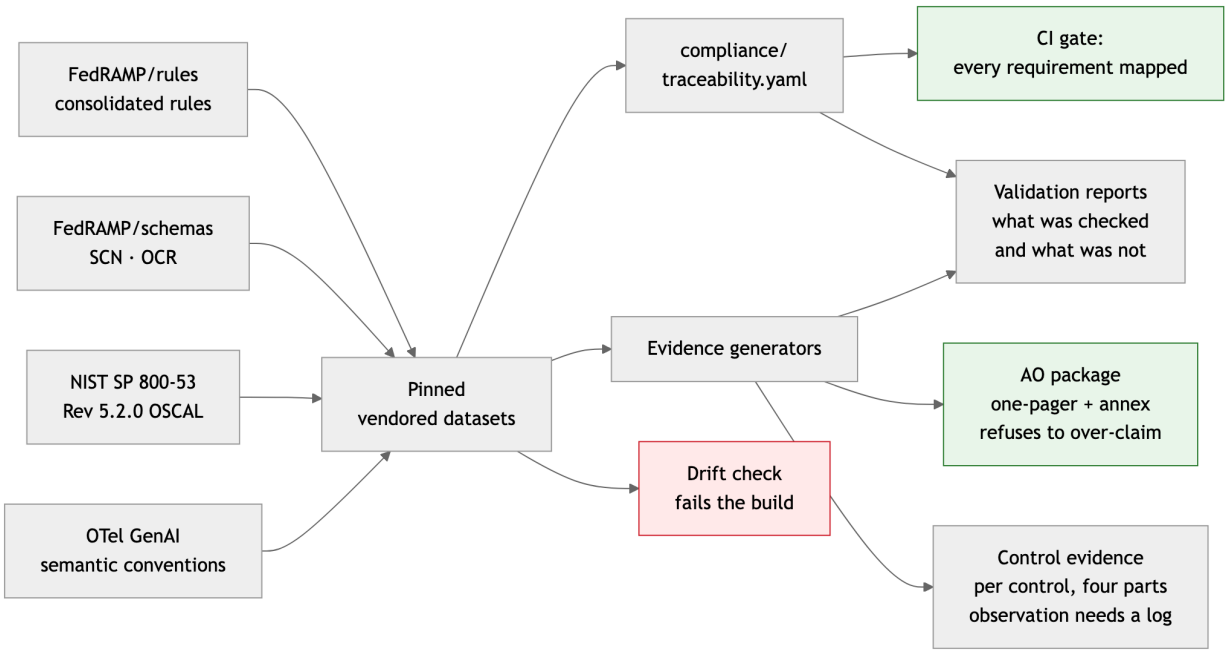
What is deliberately **not** chained: requests that establish no principal at all. They are counted in `lowrisk_admin_unauthenticated_total`. An unauthenticated write path into an unbounded in-memory store is a denial-of-service surface, and the record would name nobody. The residual — a brute-force attempt is a count, not an attributed trail — makes alerting on that counter a compensating control rather than monitoring hygiene.

What never enters the pipeline: prompt and response content, matched values, and token maps. Default telemetry is token counts, model identifiers, latency, stream metadata, and **categorical** violation types. Enabling content capture changes the sensitivity classification of the sink, so it requires acknowledgement in configuration and is surfaced in generated evidence for the AO to see (REQ-TEL-4).

---

6. Where the compliance data comes from

Control and KSI mappings are not hand-maintained prose. They are generated from pinned, drift-monitored upstream sources — the same discipline used in the `rmfcoach` repo.



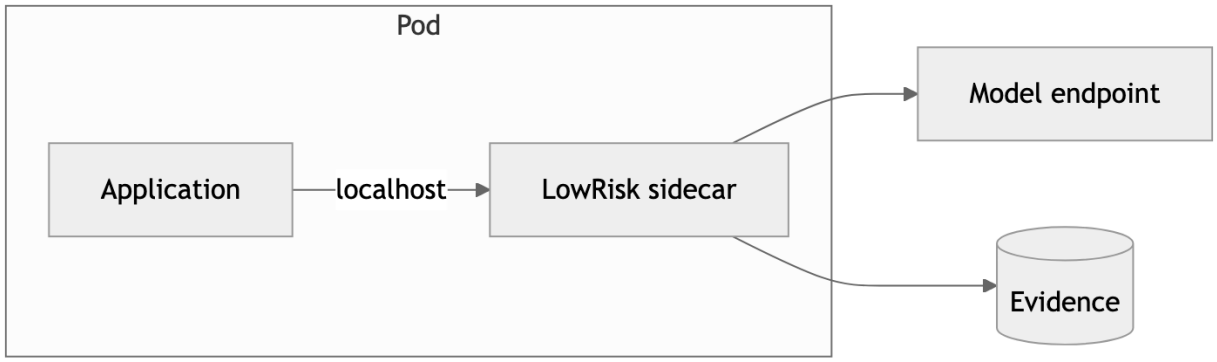
Every pin records the upstream commit. The drift check fails the build when upstream moves without review — mandatory rather than optional for the OTel GenAI conventions specifically, because they carry Development stability and moved repositories in June 2026.

A verification method is chosen, and its evidence artefact exists — or the build fails. `gen_traceability.py` used to assign a method by requirement *group*, which gave sixteen requirements `schema-validation` in a repository containing two schemas, five of them marked `implemented` while the artefact that method names was written by nothing (decision 0051). `scripts/gen_validation_report.py` now writes the two evidence artefacts that are real — `compliance/generated/schema-validation-report.json` and `artifact-gate-report.json` — each recording the artefact and schema digests, the rules applied, and **what the validation does not reach**; each also lists the requirements claiming the same method with no validation behind them. `make check` fails if an `implemented` requirement claims one of those methods and has no registered validation, so a status change now forces the choice. The EV and TEL groups carry no default method at all.

---

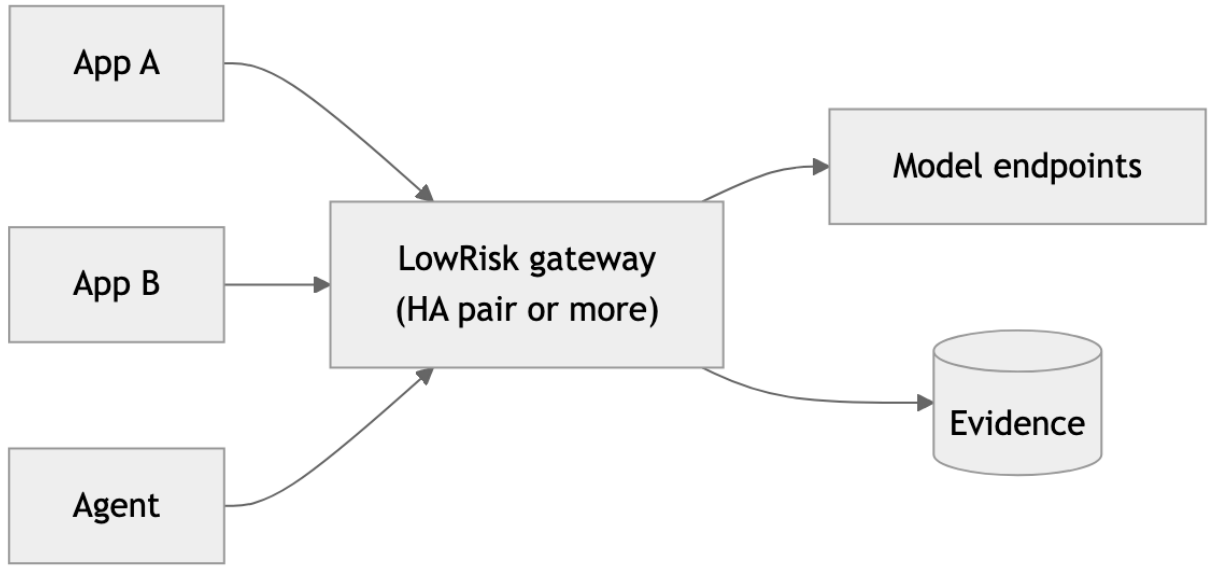
7. Deployment topologies

7.1 Sidecar — per-workload, tightest blast radius



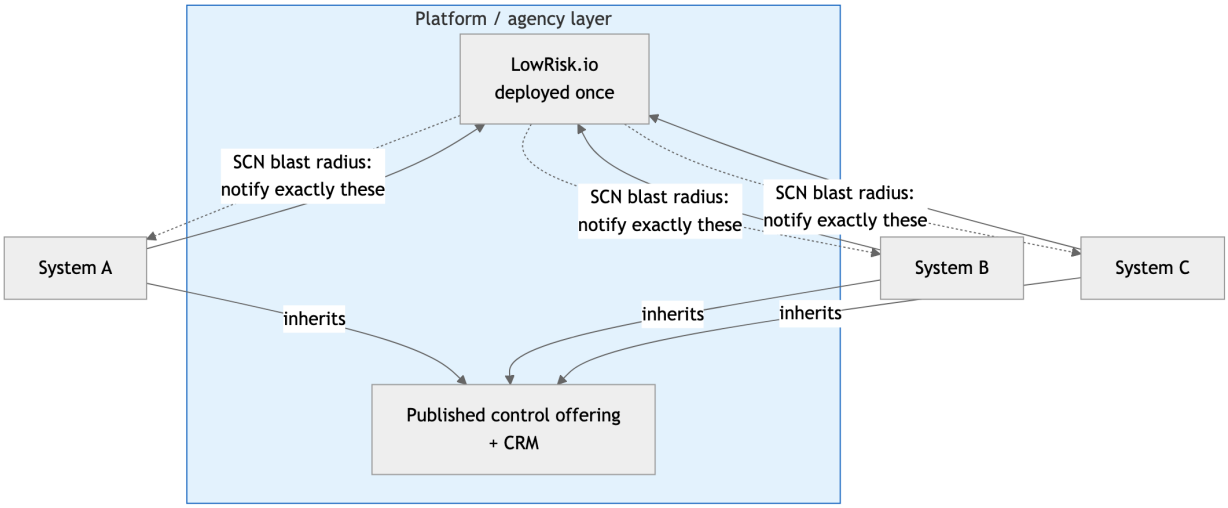
Strongest isolation and per-workload policy. Costs one instance per pod. Use where a single high-sensitivity system needs its own posture.

7.2 Gateway — per-boundary, the common case



One enforcement point, one inventory, one evidence stream. Stream classification does the per-application differentiation that a sidecar would do by placement.

7.3 Common Control Provider — the highest-leverage pattern



One deployment, one authorization conversation, N inheriting systems. The control offering is versioned and machine-readable; the CRM derives from it mechanically; inheritor enumeration is what makes SCN notification precise rather than a broadcast.

What decision 0053 built, and the line it did not cross. The offering (`oscal-component-definition.json`) now carries an **`offering-content-version`** — a digest of its dispositions, deployment-agnostic and stable under `--check` — so an inheriting SSP can cite *exactly* what it inherited and a change to that moves the version. The CRM (`customer-responsibility-matrix.md`) is the **per-inheriting-system** view: it overlays the deployment profile's `crmDelta` onto the offering, marking each hybrid row's **Source** as

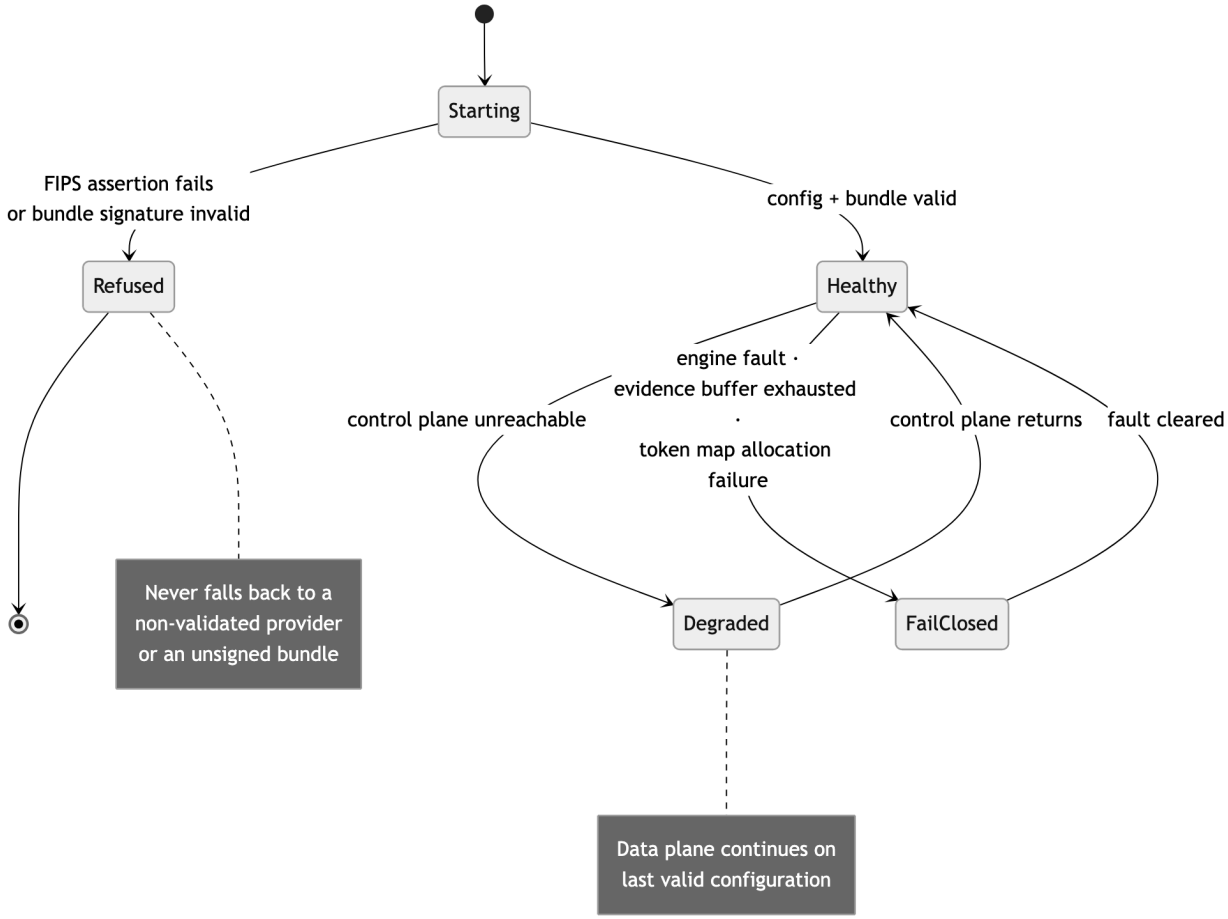
component (a universal prerequisite) or *this deployment* (assigned to this agency specifically), and it **refuses to generate** when the offering and the profile disagree about who is responsible for a control — the check that forced AC-4, AU-2 and SI-4(4) from "fully inherited" to hybrid. **Inheritor enumeration is deliberately NOT here**: which systems inherit — the blast radius the dotted line above draws — is REQ-DEP-6, the deployment pattern, not the artefact. This is a product concept, not a deployment diagram — see PRD §4.2.1 and REQ-DEP-6.

---

8. Failure behavior

Fail-closed is the default everywhere it can be. The full table is PRD §7.3; the design consequence is that **every failure path must be able to emit an evidence record**, which means the evidence recorder cannot depend on the components that fail.

The acknowledgement spine. Fail-closed-by-default is one instance of a broader rule: every toggle ships at its safest value — fail-closed, minimize, deny-by-default — and every move away from it in a posture-loosening direction is meant to be an attributed, recorded act, not a silent line in a YAML file. Four toggles enforce this as a hard mechanism today — a posture-loosening value refuses to load unless a named accepter and a reason are supplied: `fail_open_acknowledged_by`, `evidence_log.damaged_acknowledged_by`, `admin.recency_waiver`, and — since decision 0068 — `compatibility_mode` (the model-shaped denial). Decision 0068 catalogs the ~30 admin-decidable toggles in `docs/governance-toggles.md` — each with its safe default, governance driver, mapped control, and whether it is under the idiom or a candidate — and proposes **governance profiles**: named posture bundles per regime (FedRAMP-High, HIPAA, PCI-DSS, Commercial-minimal) an admin selects as a starting point, with the profile and every deviation recorded in evidence (REQ-GOV-1/2, **planned**). Generalising the idiom across the catalog is REQ-GOV-1, and decision 0070 took its first step: the four points now route through one shared validator (`require_acknowledgement`), so what counts as an acknowledgement — a non-blank, non-placeholder name, and a reason where the toggle asks for one — is decided in one place rather than four that had drifted (fail-open accepted a blank name; compatibility mode never blank-checked), and a central registry test asserts every one of them refuses a blank. It is still enforced only at those four points, not universally; bringing the catalog's candidate toggles under the idiom is the rest of REQ-GOV-1.



---

9. Design constraints worth stating

These are the non-obvious ones that shape the code. Each traces to a requirement.

Constraint	Consequence	Source
Token map is owned by exactly one stream session	No global cache, no shared pool, no cross-task handoff. Enforced by ownership in the type system where possible, and by a concurrency test that exercises connection reuse and task migration	REQ-TOK-2, SC-4
Detection is streaming	Detector interface is a sliding-window trait, not a whole-payload function	§3, REQ-PERF targets
Evidence recorder must survive component failure	It cannot depend on the detection engine, the policy runtime, or the control plane	REQ-FAIL-2

Constraint	Consequence	Source
No content in telemetry by default	The evidence schema physically has no field for it on the default path — not a flag that could be flipped by misconfiguration	REQ-TEL-3, REQ-TEL-5
Air-gap is a first-class mode	No license server, no vendor callback, no outbound call except configured endpoints and customer sinks. This forecloses usage-based licensing designs	REQ-OPS-4, REQ-DEP-4
Runs under PSS Restricted	Non-root, read-only rootfs, all capabilities dropped. This is precisely what disqualified the eBPF design, so it is a hard requirement rather than a preference	REQ-SEC-16
A generated artefact may not out-claim its own sources	The AO package's limitations section is derived from the `NOT COVERED` clause of the same statements its coverage section reads, so the two cannot disagree; and `gen_ao_onepager.py` REFUSES to emit if its own output contains assertion language. Adding a coverage claim lengthens the limitations list automatically	REQ-EV-7, REQ-EV-9, CA-6, RA-3, decision 0046
Open-core line is architectural	Module boundaries follow the Apache-2.0 / commercial split from the first commit — a codebase cannot be cleanly split along a line drawn later	REQ-DEP-7/8, D2
An enforcement flag must have something that enforces	`compliance/signing-boundary.json` carries `enforced`, and `ci.yml` invokes the check on every build even while it is `false` — so flipping the field is one edit rather than two, and cannot silently mean nothing. `scripts/check_ci_policy.py` R8 fails the build if the two are separated	REQ-SEC-12, SR-11, decision 0039

Constraint	Consequence	Source
The content core is protocol-agnostic; the LLM is an adapter	Detection/policy/allowlist/evidence/redaction/held inspect bytes and record events without parsing LLM structure. The `forward_upstream` `POST /` coupling is un-hard-coded (decision 0078): the forward path preserves the client's method/path/headers and inspects those channels fail-closed. The first MCP per-tool policies are built: a per-stream tool allowlist (REQ-MCP-2, decision 0080) governing which `tools/call` may cross, tool-supplied-URL egress filtering (REQ-MCP-4, decision 0081) refusing a tool a URL into a private/link-local/metadata range, per-tool parameter validation (REQ-MCP-3, decision 0082) refusing an allowlisted tool arguments it was not declared to receive, and token-audience inspection (REQ-MCP-5, decision 0083) refusing a bearer token minted for a different audience — all parsing the JSON-RPC envelope (or, for the token, the Authorization header) to decide while forwarding the raw request and refusing ambiguity. The LLM-specific surface left is SSE reassembly and completion tokenization; REQ-MCP-6 (telemetry) is the next slice.	REQ-MCP-1, decisions 0075/0078/0080/0081/0082/0083

---

10. Component map

Seven crates and two applications. **Every one appears here, and `scripts/check_documentation.py` DOC1 fails the build if a workspace member is missing from this file.** That gate exists because two crates — `lowrisk-enrichment` and `lowrisk-authz` — had drifted out of this document entirely and nobody noticed, which is the prose form of the defect decisions 0018 and 0019 are about: the artefact that reports was well-formed and the thing that would have caught the omission did not exist.

Crate	Licence	Plane	Wired?	What it is
`lowrisk-proxy`	Apache-2.0	data	yes	The binary. Ingress and TLS (`tls.rs`), stream classification (`classify.rs`: header + route match, deny-by-default default — decision 0054), request lifecycle (`service.rs`), the data-plane HTTP serving surface — the accept loop, the request handler, upstream re-origination and response streaming (`server.rs`, a **library** module so a `tests/` integration test can drive the wire path; `main.rs` calls `server::serve` with `server::handle_data` — decision 0065), streaming response inspection (`streaming.rs`), client-certificate identity (`identity.rs`), configuration loading with an `UNSUPPORTED` registry (`loader.rs`), admin endpoints and metrics

Crate	Licence	Plane	Wired?	What it is
`lowrisk-detect`	Apache-2.0	data	yes	Layer 1 deterministic validators (`validators.rs`) and the analyzer registry. Checksum and format rules, credential and key detectors, overlap resolution, confidence scoring
`lowrisk-policy`	Apache-2.0	data	yes	The four enforcement modes and the policy decision. (Stream CLASSIFICATION lives in `lowrisk-proxy` — `classify.rs/config.rs/loader.r` — not here; `lowrisk-policy` has no stream, profile, or sensitivity type. Corrected in decision 0054.) `contract.rs` is the compiled source of truth for the denial contract — `docs/integration/` is generated from it, so the docs cannot drift from the binary
`lowrisk-evidence`	Apache-2.0	evidence	yes	Hash chain and recorder (`sha256.rs`), tenant binding inside the chain (`tenant.rs`), clock identity and skew (`time.rs`). Every record is constructed through one function, so hardening it cannot miss a caller

Crate	Licence	Plane	Wired?	What it is
`lowrisk-enrichment`	Apache-2.0	evidence	yes, out-of-band	Layer 3. Sampled, asynchronous NER behind a pluggable recognizer (`worker.rs`); derived signals including LS-5 repeated-block detection (`signals.rs`). **Advisory: it cannot block**, and its records carry a `Provenance` marker saying so.
`lowrisk-authz`	Apache-2.0	control	**partial**	Spike D6 is why Roles, tenant scoping, second-factor evidence and policy (`lib.rs` — see §2.1) and access review (`review.rs`). `Authorizer::check` gates the two privileged admin endpoints as of 2026-07-27 — the crate's first runtime effect. Everything else (tenant derivation, cross-tenant checks, access review, deprovisioning) still has no caller and is named individually in `compliance/disconnected-allow.`

Crate	Licence	Plane	Wired?	What it is
`lowrisk-enterprise`	**commercial**	evidence	yes	Evidence bundle assembly (`bundle.rs`), the operator-facing period export (`export.rs`), CRM generation. **SCN drafting is NOT in this crate** — it is `scripts/gen_sia.py`, which derives the notification from the declared system profile rather than from anything the data plane holds; the impacted-control set is the change's own, not the component's whole mapping (0050). `EvidenceBundle::try_build` validates cross-field consistency and refuses to emit an inconsistent bundle; it takes a `Period` carrying explicit bounds, so a bundle cannot be asked for without saying what it covers. The `evidence-export` binary reads a durable log through a **read-only** handle — the reporting path cannot alter the evidence it reports on
`apps/console`	**commercial**	control	partial	Control plane UI. Currently has nothing to call — it is blocked on the same admin API as `lowrisk-authz`
`apps/testbench`	**commercial**	control	yes	Policy test bench

The `Apache-2.0` / `commercial` split is load-bearing and structural, not a header comment: a crate cannot depend upward from open to commercial, and `scripts/check_license_boundary.py` fails the build on a violation.

The `Wired?` column is the one to read carefully. A crate that is present, compiles, and passes its tests is **not** thereby part of the running system, and this table is the only place that distinction is written down.

Repository layout

```
.
  crates/                      # the seven above
  apps/
     console/                  # control plane UI                                  [commercial]
     testbench/                # policy test bench                                 [commercial]
  compliance/
     traceability.yaml         # requirement → verification → evidence → controls
     control-statements.yaml   # HOW each control is satisfied - human-written
     disconnected-allow.yaml   # every unwired guard, with a written reason
     signing-boundary.json     # the commit after which signatures are required
     pinned/                   # vendored KSI, control catalog, FedRAMP schemas
  deploy/
     helm/                     # Kubernetes, PSS-Restricted by default
     compose/                  # single node + air-gap
     siem/                     # generated SIEM content packs
  docs/
     architecture.md           # this file
     user-manual.md            # what an operator reads to run it
     decisions/                # design documents - numbered decision records
     verification-discipline.md
     integration/              # generated from the compiled contract
  scripts/                     # gates (check_*.py, check_*.sh) and generators (gen_*.py)
  .github/workflows/           # ci · release · compliance
```

The five artefact types the project rule names map onto this tree as: `docs/user-manual.md`, `docs/decisions/`, `docs/architecture.md`, `lowrisk.io.prd.md`, and `compliance/control-statements.yaml`. `scripts/check_documentation.py` DOC5 fails the build if any is absent.

DOC5 is a real question about four of them and no question at all about the fifth, because `docs/decisions/` is a directory and a directory does not stop existing. Decisions 0043 and 0045 landed on `main` with no record and no index row while every gate stayed green, which is decision 0018's subject inside the gate written to enforce 0020. Two mechanisms cover it now:

	reads	catches
`check_documentation.py` **DOC10**	the tree	a hole in the numbering, a record with no index row, a row resolving to nothing, a number claimed twice. Retrospective — it is what caught 0043 and 0045, and it cannot fire on the commit that skipped a number

reads	catches
`scripts/check_decision_record.py` git	a commit that changes the system and adds no indexed record, refused at push time. Prospective — it reads the range a push introduced and cannot revisit history

Both run in `make check` and in `compliance.yml`. The git one exits **2** rather than 0 when it cannot resolve the range it was given — a shallow checkout is the usual cause — because a gate that answered by examining `HEAD` alone would report success over commits it never read (decision 0049).

Related

- `docs/verification-and-evidence.md` — how each requirement here is proven
- `docs/deployment.md` — how to run it, including air-gapped
- `compliance/traceability.yaml` — the machine-readable matrix
- PRD v4.1.0 §2 (threat model), §7.3 (failure modes), §10 (product security)