ACCESS CONTROL LABEL eXtended

ACLX™ Specification · Version 0.6 Draft

A machine-readable governance and enforcement labeling standard for AI-generated output

Status	Draft - Pre-Release
Version	0.6
Classification	UNCLASSIFIED / WORKING DRAFT
Author	Chris Hunt
Last Updated	July 2026

31 Knowledge Integrity

The ACLX enforcement layer (Layers 3 and 4) answers one question about an AI response: who may receive this answer, and how much of it? The Knowledge Integrity capabilities extend the same engine to a second, adjacent question the classification layer never asked -- can this response be trusted, and is the knowledge behind it sound? An output can pass every access check and still be wrong, fabricated, or poisoned: assembled from a publicly-editable wiki, contradicting an authoritative reference, produced by a model whose behaviour has quietly drifted, or asserting something no other model would.

These four capabilities build directly on the source-provenance model (Section 22), groundedness control (Section 23), and standards-corpus grounding (Section 24). Where those sections made "a synthesized output can be more sensitive than any single source" an enforced property, Knowledge Integrity makes "an output is only as trustworthy as the sources -- and the model -- behind it" an enforced property.

> **The Question Knowledge Integrity Answers**
> An output is only as trustworthy as the sources -- and the model -- behind it. Knowledge Integrity makes the provenance and soundness of the knowledge a first-class enforcement input. The maximum action any Knowledge Integrity signal can produce is ESCALATE (route to human review) -- none can hard-block -- consistent with ACLX's conservative-by-default posture.

Design invariants (all four capabilities). Each capability holds to the same contract, so each is safe to enable independently:

- **Additive schema.** New optional request fields default to absent; new label content nests additively under provenance.* and is omitted entirely when the feature is off. The schema remains acl_version 0.5-compatible.
- **Feature-flagged, default OFF.** Each capability is gated by its own environment flag. With all flags off, scoring, the OPA input, and the response label are byte-for-byte identical to the baseline gateway.
- **No new decision values.** The decision set is unchanged (ALLOW | REDACT | BLOCK | ESCALATE). New OPA packages reuse the existing escalate / flag / pass effects; the maximum effect of any Knowledge Integrity signal is ESCALATE.
- **Fail-safe.** Absence of a signal never escalates or blocks. An unavailable LLM checker, a peer that times out, or a corpus that is not loaded degrades to "no finding," never to a denial.

Status: these capabilities are built, tested, and available behind their feature flags. They preserve the existing enforcement contract and are additive, so a deployment adopts them one flag at a time.

31.1 Trusted Knowledge Zones -- Graded, Zone-Based Source Trust

Flag: KNOWLEDGE_INTEGRITY_ENABLED (default off). The baseline gateway scored a source as trusted or not by a single heuristic -- does it carry a provenance owner? Trusted Knowledge Zones replace that binary with a graded, client-declared provenance tier: an ordered ontology of ten zones from hardened government down to publicly-editable content, each with a numeric trust weight. Government tiers are graded by data-poisoning risk (federal is the most hardened, local the softest); public tiers are graded by mutability and reputability (an immutable cited reference ranks above an official site, which ranks above open, editable content); curated non-government, mission, sensor, and human tiers sit between them.

Tier group	Zone	Weight	Graded by
Government	gov_federal	1.00	Data-poisoning risk / security posture -- federal is the most hardened government source.
(all above public)	gov_state	0.93	State government source.
	gov_local	0.86	Local government -- the softest government target.
Curated non-gov	human_approved	0.80	Human-vetted content.
	verified_org	0.72	Verified organizational source.
	mission_specific	0.64	Curated mission corpus.
	sensor_feed	0.55	Machine-attested telemetry.
Public (graded)	public_reference	0.48	Mutability / reputability -- an immutable cited / academic reference is the highest public tier.
	public_official	0.40	An official site.
	public_open	0.30	Publicly-modifiable UGC (wikis, forums) -- the lowest tier overall; an untagged source is treated as this tier.

- **Zone-weighted TrustEngine.** On the zoned path the traceability score is min(0.65 + mean(zone_weights) x 0.35, 1.0) -- the same formula shape as the legacy score, but each source contributes its zone weight instead of a 1/0 owner flag. The legacy formula is preserved bit-for-bit for any request that carries no zones.
- **acl.integrity policy.** Grades the content's sensitivity against the lowest zone that supports it, reusing escalate / flag / pass. HIGH or CRITICAL content whose sole support is publicly-modifiable escalates; an

immutable public_reference or gov_federal source corroborates rather than triggers, so sole support by them never escalates.

- **trust_pedigree label.** An additive summary (zones, zone_counts, weighted_trust, corroborated, highest / lowest tier) records the source pedigree in the signed label, emitted only on the zoned path.

31.2 Per-Claim Assertion Records and Corpus Corroboration

Flag: ASSERTIONS_ENABLED (default off); corpus signing: CORPUS_SIGNATURE_REQUIRED. The faithfulness detector (Section 23) already decomposes a response into claims and flags those unsupported by the provided sources. This capability promotes that decomposition into first-class assertion records and adds a second line of defense: corroborating model-parametric (ungrounded) claims against a governed corpus of authoritative facts.

- **Assertion records.** Per material claim, the detector emits { claim, supporting_source_ids, materiality }; each is enriched with the zones of its supporting sources, a corroboration_count, and a confidence band, and rides additively in provenance.assertions on the signed label.
- **Corpus corroboration.** For each material claim that no source supports, the corroboration corpus retrieves relevant entries from governed reference corpora (the ABA / BACB and HIPAA corpora ship as examples) and a batched LLM entailment check (with a deterministic fallback) decides whether an authoritative entry contradicts the claim. Regulatory entries (CFR, GINA, HIPAA, BACB) map to the gov_federal authority zone.
- **INCONSISTENT_WITH_AUTHORITATIVE_SOURCE.** A contradiction with an authoritative-zone entry emits this finding via the guarded acl.corroboration package, which escalates. Absence of corroboration is only a confidence annotation, never a gate -- authoritative-zone corroboration is a secondary, additive check, not a truth oracle.
- **Signed, versioned corpora (poisoned-RAG defense).** Each corpus carries a version and content snapshot_id and is signed offline with Ed25519 using a trust root distinct from the runtime label key. The gateway verifies at load and, when CORPUS_SIGNATURE_REQUIRED is on, rejects an unsigned or tampered corpus, so a poisoned reference set cannot silently drive or suppress a finding. The corpus version that corroborated a claim is recorded in the label for audit.

31.3 Model-as-Source Trust and Drift Detection

Flag: MODEL_TRUST_ENABLED (default off); signing: MODEL_TRUST_SIGNATURE_REQUIRED. The generating model is itself a source of knowledge -- arguably the least inspectable one. This capability gives each governed model a trust record and lets a drift downgrade flow through the existing TrustEngine and acl.integrity path, with no new enforcement machinery.

- **Model as a graded source.** A client declares the generating model via the additive ai_response.model. When enabled, the model is injected as a synthetic, text-less, zone-bearing source whose trust record maps to a zone: healthy to verified_org, degraded to public_official, quarantined to public_open. A downgraded model therefore lowers the traceability score and -- when it is the sole support for HIGH or CRITICAL content -- escalates via the existing rule. It never blocks. Its status is recorded in provenance.model_trust.
- **Signed registry.** The model-trust registry holds per-model records signed offline (distinct trust root); the latest record per model wins (monotonic -- a re-bless supersedes a downgrade). MODEL_TRUST_SIGNATURE_REQUIRED ignores unsigned or tampered records, so a poisoned registry can neither silently down- nor up-grade a model.
- **Offline drift job.** A scheduler-driven batch service (never on the request hot path) runs a fixed, signed canary set through each governed model, diffs the answers against the model's signed, versioned baseline (LLM-judge equivalence with a deterministic fallback), and blends in a passive signal from the hash-chained audit stream (per-model escalation, inconsistency, and consensus-anomaly rates). On material drift it writes a signed downgrade and alerts. Downgrades are monotonic -- a model returns to healthy only via a freshly captured, signed baseline.

31.4 Multi-Model Consensus -- A Secondary Anomaly Signal

Flag: CONSENSUS_ENABLED (default off). Consensus probes a small panel of peer models on the response's material claims and flags when a majority contradict the primary. It is explicitly a secondary signal, with its limitation stated up front.

> **Consensus is an anomaly flag, not ground truth.**
> Major models share training data, so consensus detects idiosyncratic tampering -- a poisoned fine-tune or a compromised local model diverging from its peers -- not ecosystem-wide narrative poisoning. Authoritative-zone corroboration (Section 31.2) remains the primary defense; consensus only ever raises scrutiny.

- **Claim-based; reuses assertions.** Peers judge the same material, model-parametric claims (corroboration-failed ones first), so consensus depends on ASSERTIONS_ENABLED and needs no re-prompting.
- **Tiered and cost-gated.** It runs only when triggered -- HIGH or CRITICAL sensitivity, or a corroboration failure -- and only when there are claims to check: at most a few peers, a capped number of claims, one batched call per peer, probed in parallel with a per-peer timeout.
- **Flag-only and fail-safe.** The guarded acl.consensus package emits flag (anomaly) or pass (consistent) -- never escalate or block, and a consensus anomaly can never change the final action. Below a quorum of responding peers the run is unavailable and flags nothing.
- **Independence caveat surfaced.** Peers are configured provider-agnostically, but where the platform permits only one provider the peers share a provider family, weakening independence. The summary carries same_provider and a caveat string into provenance.consensus and the label so reviewers know how much weight the flag deserves.
- **Feeds model trust.** A per-model consensus-anomaly rate is an additional passive input to the Section 31.3 drift job -- a model whose claims peers repeatedly contradict trends toward a drift downgrade over time, with no hot-path enforcement.

How they compose. Graded provenance (31.1) and model-as-source trust (31.3) feed the TrustEngine and acl.integrity; per-claim corroboration (31.2) feeds acl.corroboration; consensus (31.4) feeds acl.consensus as a flag only. Authoritative-zone corroboration and graded provenance are the primary, evidence-grounded signals; consensus is deliberately secondary. Nothing here can hard-block a response -- the strongest action any Knowledge Integrity signal produces is routing to a human reviewer.

1 Executive Summary

The Access Control Label eXtended (ACLX) is a machine-readable governance and enforcement labeling standard for AI-generated output. As AI systems increasingly synthesize content from heterogeneous sources -- crossing regulatory boundaries, sensitivity levels, and distribution authorities -- existing access control models fail at a critical point: the output boundary.

Traditional identity and access management (IAM) controls govern what data an AI system may retrieve or process. They do not govern what that system may generate or how the generated output should be handled downstream. ACLX addresses this structural gap.

> **Core Problem**
> Traditional IAM enforces policy at the request boundary. AI-generated output is a new artifact -- unlabeled, unnegotiated, and outside the governance perimeter of any system that supplied the input data.

ACLX solves this by treating every AI-generated response as a first-class governance object. It normalizes security, privacy, safety, legal, and regulatory control assertions into a unified, machine-readable structure evaluated against the requesting identity by a policy engine such as Open Policy Agent (OPA).

Design Intent

- **Identity-aware:** Enforcement outcomes depend on who is asking, not only what was generated.
- **Domain-agnostic:** A single schema covers HIPAA, CUI, GDPR, ITAR, IP, and content-safety domains simultaneously.
- **Externalized:** Governance logic lives outside the model -- in the ACLX enforcement layer -- preventing model-level bypass.
- **Auditable:** Every label, decision, and identity assertion is logged and explainable.
- **Composable:** A single response may carry multiple simultaneous domain labels with independent enforcement logic.

2 Five-Layer Adaptive Access Control Framework

ACLX does not operate in isolation. It is Layer 3 of a five-layer adaptive access control framework designed to close the structural gap between what traditional IAM controls protect and what AI systems produce. Understanding where ACLX sits within this framework is prerequisite to understanding how it integrates with cloud DLP services, policy engines, identity providers, and audit systems.

> **The Core Gap**
> Traditional IAM controls (RBAC, ABAC, Zero Trust) enforce policy at the request boundary. They correctly protect labeled input data. But AI generates new, unlabeled output data -- and no current IAM framework defines how to classify, label, or enforce policy on that output. HIPAA does not address it. DoD 5200.01 does not address it. GDPR has partial coverage. The frameworks assume content arrives pre-labeled. It does not.

2.1 Layer Reference

Layer	Function	Product Role	Coverage and Gap
Layer 1	**Identity Verification**	IDP / OIDC / SCIM	Authentication, MFA, session management, role and clearance claims issuance. ABAC augmentation for AI-specific contexts. Gap: claims issued at login only -- no visibility into what the model synthesized post-session.
Layer 2	**Context Analysis**	AuthZ Engine / Fine-Grained AuthZ	Attribute-based policy at request time. Evaluates: can this identity perform this action on this resource? Gap: evaluates the request, not the response. Purpose-of-use is lost at the response boundary.
Layer 3	**Output Classification**	ACLX Engine + DLP / Ontology / LLM Eval	Real-time detection, classification, and labeling of AI-generated output. Generates the ACLX label. Gap that ACLX fills: no existing DLP product hooks into live LLM output streams or detects synthesis sensitivity.

Layer 4	**Policy Enforcement**	OPA / Cedar / AuthZEN	Decoupled policy evaluation consuming the ACLX label and identity attributes. Returns: ALLOW, REDACT, BLOCK, or ESCALATE (PARTIAL_ALLOW and QUARANTINE are reserved). Gap: ready to evaluate -- it just lacks Layer 3 evidence. ACLX is the integration point.
Layer 5	**Audit and Governance**	IGA / SIEM / SCITT	Access certification, entitlement reviews, AI output decision logging, regulatory reporting. Gap: captures access decisions -- not AI output decisions. ACLX audit records close this gap.

Layers 1 and 2 already exist in most enterprise environments. The gap is Layer 3 -- the output classification layer. Layers 4 and 5 are also typically present but are missing the Layer 3 evidence that would make them useful at the AI output boundary. ACLX is the integration contract between Layer 3 and Layer 4.

2.2 What AI Defeats in Each Layer

AI-generated output creates a specific failure mode at each layer of the framework. Understanding these failures motivates the ACLX design:

- **Layer 1 failure -- Agent actors bypass identity entirely:** Agentic AI systems operate with no IDP session, no claims, and no human review. Machine-to-machine communication never triggers authentication checks. The agent retrieves, synthesizes, and forwards data across system boundaries before any human sees the output. ACLX addresses this by requiring origin attestation in the ACLX label regardless of whether the requestor is a human or an agent.
- **Layer 2 failure -- RBAC checks access, not output:** RBAC verifies what the identity is authorized to query. It does not evaluate what the model returned. A Billing role that is authorized to query a patient database passes every RBAC check -- and receives a response containing full clinical notes, diagnosis history, and clinical trial status that no billing workflow requires. ACLX addresses this through purpose-bound output controls enforced at Layer 4.
- **Layer 3 failure -- Synthesis creates sensitivity that never existed in source data:** Existing DLP products classify data that exists at rest or in transit. AI synthesis creates new data at inference time -- a novel output that was never stored anywhere and was never labeled by any source system. No traditional DLP tool intercepts a live LLM output stream and classifies synthesis-created sensitivity. This is the gap ACLX and Layer 3 fill.
- **Layer 4 failure -- The policy engine has no evidence to evaluate:** OPA and Cedar are ready to evaluate output-time policy. They lack the structured evidence -- the ACLX label -- that would tell them what sensitivity the output contains. Without Layer 3, Layer 4 is inert at the output boundary. ACLX is the evidence contract Layer 4 has been waiting for.
- **Layer 5 failure -- Audit captures access decisions, not output decisions:** IGA systems log what identities were authorized to access. They do not log what the AI model actually produced, what sensitivity was detected, or what enforcement action was applied to the output. ACLX audit records close this gap by writing output-time decisions into the same audit infrastructure that governs the rest of the access control lifecycle.

2.3 One Framework, Three Regulatory Regimes

The five-layer framework applies uniformly across regulatory regimes. The following table shows how each layer maps to the specific compliance requirements of HIPAA, DoD CUI (5200.01), and GDPR. This is the same technical architecture producing different regulatory outcomes -- not three different frameworks.

Layer	HIPAA	DoD CUI (5200.01)	GDPR
Layer 1 Identity	User authentication Role validation	Role + need-to-know Clearance verification	Data controller ID Processor accountability
Layer 2 Context	Purpose of use Minimum necessary assessment	Distribution Statement Need-to-know evaluation	Legitimate interest Purpose limitation
Layer 3 Classification	PHI detection Compilation risk scoring	DoD Distribution assignment Synthesis sensitivity labeling	Sensitive data identification Special category detection
Layer 4 Enforcement	Minimum necessary Redaction / block on violation	Distribution control Block / redact on elevation	Data minimization Right to object enforcement
Layer 5 Audit	Accounting of disclosures Breach audit trail	Oversight reporting Classification decision log	Processing records Data subject request log

3 ACLX within the AEGIS Framework

ACLX does not stand alone. It is the runtime enforcement plane of the broader AEGIS framework. AEGIS is the governance authority: it decides who and what may act, registers agents and issues their attestations, owns data-governance policy, holds signing keys, and aggregates audit. ACLX is the enforcement runtime that applies that governance to each AI-generated response at the output boundary. Section 2 placed ACLX as Layer 3 of the five-layer access-control model; this section makes the platform relationship explicit and divides responsibility cleanly between the two.

> **A note on terminology**
> ACLX (Access Control Label eXtended) is the enforcement plane -- the runtime acting as the Policy Enforcement Point (PEP). AEGIS is the governance plane above it: the central authority that authors policy, issues attestations, and custodies keys. The names are independent -- ACLX is the labeling and enforcement standard; AEGIS is the framework it runs beneath -- and the roles are distinct: AEGIS sets the rules, and ACLX applies them at the output boundary.

3.1 The Three Planes

AEGIS and ACLX form a layered control stack with a clean separation between authoring governance centrally and enforcing it at the point of delivery. Policy bundles and public keys flow down when a link is available; signed audit flows up when the link returns. No call-home sits on the decision path -- the enforcement plane is designed to run autonomously (Section 27).

```
Governance plane (AEGIS)    policy authoring | agent registry | key custody | audit
      |   down: signed policy bundles, public keys, agent attestations & trust
      v
ACLX enforcement plane      classify -> trust -> enforce -> label -> explain -> audit
      |   down: signed Content Label (passport) + decision      up: signed audit
```

```
        v
Delivery plane              users | agents | MCP servers | APIs | applications
```

Plane	Responsibility	Relationship to ACLX
Governance (AEGIS)	Authors and promotes policy, registers agents and issues attestations, owns data governance, custodies and rotates keys, aggregates audit.	Sits above ACLX; supplies policy bundles, public keys, and attested trust inputs.
ACLX (enforcement)	Classify, score trust, resolve and enforce the action, issue the signed label, explain, and audit -- at the output boundary.	Is the runtime; consumes governance inputs and emits a signed Content Label and audit.
Delivery	Human users, agents, MCP servers, APIs, and applications that receive the governed response.	Receives the response plus its verifiable Content Label (Section 16); below ACLX.

3.2 Division of Responsibility

The split is deliberate: governance is authored once, centrally, and enforced everywhere, locally. Each row below names a concern and shows which plane owns which half of it.

Concern	AEGIS governance plane	ACLX enforcement plane
Policy	Authors and promotes signed policy bundles.	Evaluates the last-known-good bundle on every request.
Identity & agents	Registers agents; issues AEGIS attestations.	Consumes the attestation, or scores heuristically when absent (Section 18).
Keys	Custodies and rotates signing keys; publishes public keys.	Signs each Content Label with the on-node key; verifies upstream labels.
Classification & decision	Defines the rules.	Detects, classifies, and resolves the action at the output boundary.
Trust inputs	Supplies attested identity and trust.	Combines content provenance, agent, and chain trust into the decision.
Audit	Aggregates and reviews the signed audit stream.	Writes the tamper-evident record; store-and-forwards when disconnected.

3.3 What Crosses the Boundary

Only three things move between the planes, and the content being governed is never one of them -- it stays inside the enforcement boundary.

- **Down (governance to ACLX):** signed policy bundles, public keys, and the agent registry / attestations descend from AEGIS to each ACLX node.
- **Down (ACLX to delivery):** the signed Content Label -- the passport (Section 16) -- accompanies the governed response to the delivery plane, where any consumer can verify it without calling back.
- **Up (ACLX to governance):** the signed, hash-chained audit stream ascends to the governance plane, store-and-forwarded when a link is unavailable.

> **Why the split matters**

> Because governance is authored centrally but enforced locally, an ACLX node makes a correct, conservative boundary decision with no dependency on the cloud at decision time (Section 27). The governance plane sets the rules and holds the keys; the enforcement plane proves -- at the boundary and at the moment of delivery -- that those rules were applied to this response, for this identity.

4 Core Architectural Principles

ACLX is grounded in five foundational principles that govern both schema design and enforcement behavior:

4.1 Output-as-Object

ACLX models control assertions against AI-generated output as discrete governance objects. Generated content is not assumed to inherit the controls of its source data; it requires independent labeling and evaluation. This is the principle that makes DLP service integration possible: the output is treated as an inspectable artifact, not a passthrough.

4.2 Multi-Domain Simultaneity

A single generated output may simultaneously belong to multiple governance domains. A response synthesizing EHR data and export-controlled technical information carries HIPAA, ITAR, and potentially SAFETY domain labels at the same time. ACLX evaluates all active domains and applies the most restrictive valid action across domains.

4.3 Trust and Provenance as Enforcement Attributes

Trust is not assumed. The degree to which generated content is traceable to verified source material is a first-class enforcement attribute, not an annotation. Low traceability scores shift enforcement toward REDACT or BLOCK regardless of sensitivity level.

4.4 Identity-Aware Policy Evaluation

The same generated output may yield different enforcement outcomes for different requesting identities. ACLX labels are evaluated against identity attributes -- role, purpose, clearance, organizational affiliation -- at enforcement time. Labels do not encode decisions; they encode the governance state that enables decisions.

4.5 Auditability and Explainability

Every enforcement decision must be reproducible from the ACLX label, the requesting identity attributes, and the applicable policy. The audit record must be sufficient to explain any decision to a compliance authority without reference to model internals.

5 Canonical ACLX Structure

The ACLX object is a JSON document attached to or associated with every AI-generated response. The top-level fields establish identity for the response artifact; nested objects carry domain-specific governance assertions.

```
{
  "acl_version": "0.5",              // Schema version. Evaluated by ACLX engine for
compatibility.
```

```
  "content_id": "string",            // Stable unique identifier. Used in audit and
retrieval.
  "generated_at": "ISO-8601",        // Timestamp of generation. UTC required.
  "origin": {                        // Source system and model identity.
    "system": "string",
    "model": "string",
    "session_id": "string"
  },
  "control_domains": [],             // Array of SCDO objects. One per active domain.
  "trust": {                         // Aggregate trust level across all domains.
    "level": "HIGH | MEDIUM | LOW | UNKNOWN",
    "justification": "string"
  },
  "provenance": {                    // Source traceability model.
    "mode": "RAG | SYNTHETIC",
    "traceability_score": 0.0,
    "sources": [],
    "synthesis": {}
  },
  "handling": {                      // Aggregate handling directives across all domains.
    "action": "ALLOW | REDACT | BLOCK | ESCALATE",
    "rationale": "string",
    "applied_at": "ISO-8601"
  },
  "audit": {                         // Immutable audit record.
    "decision_id": "string",
    "policy_version": "string",
    "identity_snapshot": {}
  }
}
```

6 Standardized Control Domain Object (SCDO)

The SCDO is the atomic governance unit in ACLX. Each active control domain for a generated response produces one SCDO placed in the control_domains array. The ACLX enforcement layer evaluates each SCDO independently, then applies cross-domain resolution logic to produce a single handling.action.

6.1 SCDO Field Reference

Field	Type / Enum	Description
domain	string (required)	Top-level governance regime. E.g., HIPAA, CUI, IP, SAFETY, GDPR.
category	string (required)	Classification within domain. E.g., PHI, TRADE_SECRET, ITAR.
subcategory	string	Refined classification. E.g., SUPER_PHI, MISSILE_GUIDANCE, PROPRIETARY_ALGORITHM.
sensitivity	enum (required)	LOW | MODERATE | HIGH | CRITICAL. Drives default enforcement bias.
distribution	string	Distribution authority string. Domain-specific. E.g., TREATMENT_ONLY, D, INTERNAL.

restrictions	string[]	Explicit restrictions. E.g., MINIMUM_NECESSARY, NO_FOREIGN_NATIONALS.
handling_requirements	string[]	Required actions on enforcement. E.g., REDACTION_REQUIRED, EXPORT_REVIEW, BLOCK_REQUIRED.
confidence.score	float [0.0-1.0]	Automated confidence in classification accuracy. Used in ESCALATE thresholds.
confidence.level	enum	LOW | MEDIUM | HIGH. Human-readable alias for score band.
provenance.traceability_score	float [0.0-1.0]	Proportion of output attributable to verified source material.
provenance.sources_verified	boolean	True if all cited sources passed integrity verification.
review.required	boolean	Flags output for mandatory human review before release.
review.authority	string	Role, team, or system responsible for review. E.g., Privacy Officer, ISSO.
metadata	object	Domain-specific extension bag. Not evaluated by core ACLX engine.

6.2 SCDO JSON Template

```
{
  "domain": "string",
  "category": "string",
  "subcategory": "string",
  "sensitivity": "LOW | MODERATE | HIGH | CRITICAL",
  "distribution": "string",
  "restrictions": [ "string" ],
  "handling_requirements": [ "string" ],
  "confidence": {
    "score": 0.0,
    "level": "LOW | MEDIUM | HIGH"
  },
  "provenance": {
    "traceability_score": 0.0,
    "sources_verified": false
  },
  "review": {
    "required": false,
    "authority": "string"
  },
  "metadata": {}
}
```

Design Note

The metadata field is intentionally untyped. Domain-specific extensions (e.g., clinical_domain for HIPAA, cta for CUI/ITAR) live here and are passed through to downstream systems without ACLX engine evaluation. This keeps the core schema stable while enabling domain-specific richness.

7 Domain Examples

The following examples illustrate complete SCDO objects for each supported domain. Real-world responses will typically carry multiple SCDOs simultaneously -- see Section 13 for multi-domain composition.

7.1 HIPAA / Protected Health Information

Applies when generated output contains, references, or is derived from Protected Health Information (PHI) or Special Category PHI (Super PHI). The MINIMUM_NECESSARY restriction is the canonical HIPAA enforcement pattern.

```
{
  "domain": "HIPAA",
  "category": "PHI",
  "subcategory": "SUPER_PHI",
  "sensitivity": "HIGH",
  "distribution": "TREATMENT_ONLY",
  "restrictions": [ "MINIMUM_NECESSARY" ],
  "handling_requirements": [ "REDACTION_REQUIRED" ],
  "confidence": { "score": 0.94, "level": "HIGH" },
  "review": { "required": false, "authority": "Privacy Officer" },
  "metadata": {
    "clinical_domain": "INFECTIOUS_DISEASE",
    "contains_identifiers": true,
    "identifier_types": [ "NAME", "DOB", "MRN" ]
  }
}
```

7.2 CUI -- Critical Technology (ITAR-Adjacent)

Applies when generated output contains or synthesizes Controlled Unclassified Information under a specific CUI SP category. Distribution D restricts release to U.S. government employees and authorized U.S. persons only.

```
{
  "domain": "CUI",
  "category": "CUI_SP_CTI",
  "subcategory": "HYPERSONICS",
  "sensitivity": "CRITICAL",
  "distribution": "D",
  "restrictions": [ "NO_FOREIGN_NATIONALS", "EXPORT_CONTROLLED" ],
  "handling_requirements": [ "EXPORT_REVIEW", "BLOCK_REQUIRED_IF_UNCLEARED" ],
  "confidence": { "score": 0.88, "level": "HIGH" },
  "review": { "required": true, "authority": "ISSO" },
  "metadata": {
    "cta": "SCADE",
    "itar_related": true,
    "applicable_regs": [ "22 CFR 120-130", "EAR 15 CFR 730-774" ]
  }
}
```

7.3 Content Safety

Applies when generated output contains material that, regardless of the requesting identity's clearance, must not be released. Content safety is a hard-block domain.

```
{
  "domain": "SAFETY",
  "category": "WEAPONS_INSTRUCTION",
  "subcategory": "EXPLOSIVE_CONSTRUCTION",
  "sensitivity": "CRITICAL",
```

```
  "distribution": "NONE",
  "restrictions": [ "NO_PUBLIC_RELEASE", "NO_INTERNAL_RELEASE" ],
  "handling_requirements": [ "BLOCK_REQUIRED" ],
  "confidence": { "score": 0.97, "level": "HIGH" },
  "review": { "required": true, "authority": "Safety Review Board" }
}
```

7.4 Intellectual Property

Applies when generated output contains, reproduces, or is substantially derived from trade secrets, proprietary algorithms, or other IP-protected material.

```
{
  "domain": "IP",
  "category": "TRADE_SECRET",
  "subcategory": "PROPRIETARY_ALGORITHM",
  "sensitivity": "HIGH",
  "distribution": "INTERNAL",
  "restrictions": [ "NO_EXTERNAL_SHARING", "WATERMARK_REQUIRED" ],
  "handling_requirements": [ "WATERMARK_REQUIRED" ],
  "confidence": { "score": 0.81, "level": "HIGH" },
  "review": { "required": false, "authority": "Legal -- IP Counsel" }
}
```

7.5 GDPR / Personal Data

Applies when generated output contains or derives from personal data of EU data subjects. The GDPR domain requires that purpose limitation be enforced at output time.

```
{
  "domain": "GDPR",
  "category": "PERSONAL_DATA",
  "subcategory": "SPECIAL_CATEGORY",
  "sensitivity": "HIGH",
  "distribution": "LAWFUL_PURPOSE_ONLY",
  "restrictions": [ "PURPOSE_LIMITATION", "DATA_MINIMIZATION" ],
  "handling_requirements": [ "PURPOSE_MATCH_REQUIRED" ],
  "confidence": { "score": 0.85, "level": "HIGH" },
  "metadata": {
    "data_subject_region": "EU",
    "lawful_basis": "LEGITIMATE_INTEREST",
    "retention_days": 30
  }
}
```

8 Trust Model

The ACLX trust model quantifies the degree to which generated output can be traced to verified source material. Trust is a first-class enforcement attribute: low-trust output is subject to more restrictive enforcement defaults regardless of the sensitivity level assigned by domain classification.

Trust Level	Definition	Enforcement Bias
HIGH	ACLX-controlled retrieval; verified provenance chain.	Permit least-restrictive valid policy. No additional validation step required.

MEDIUM	Partial provenance or externally supplied citations.	Require additional validation before release. Log for audit.
LOW	Ungrounded generation or unverifiable lineage.	Bias toward REDACT or BLOCK. Escalate in regulated domains.
UNKNOWN	No provenance information available.	Deny in sensitive domains. Route to ESCALATE queue.

8.1 Trust Level Assignment Logic

- If any source in provenance.sources has sources_verified == false and is used_in_response, aggregate trust cannot exceed MEDIUM.
- If provenance.mode == SYNTHETIC (no retrieval grounding), trust is LOW unless overridden by explicit AEGIS attestation.
- If provenance.traceability_score < 0.5, trust is LOW.
- If provenance.traceability_score >= 0.85 and all sources verified, trust is HIGH.
- All other conditions yield MEDIUM.

9 Provenance and Traceability

ACLX distinguishes between grounded and ungrounded AI generation. Grounded generation (RAG mode) produces output directly attributable to retrieved source documents. Ungrounded generation (SYNTHETIC mode) may synthesize content from model weights with no traceable source -- a higher-risk condition for sensitive domains.

Traceability is an architectural property enforced by the ACLX enforcement layer, not an assumed capability of the underlying model. The model does not self-report provenance; the retrieval and orchestration layer injects provenance evidence into the ACLX label at generation time.

9.1 Provenance Object

```
"provenance": {
  "mode": "RAG",                    // RAG | SYNTHETIC
  "traceability_score": 0.91,       // Proportion of output attributable to verified
sources.
  "sources": [
    {
      "source_id": "ehr-visit-1",
      "type": "EHR",
      "label": "PHI",              // Sensitivity label inherited from source.
      "used_in_response": true
    },
    {
      "source_id": "protocol-42",
      "type": "CLINICAL_PROTOCOL",
      "label": "UNCLASSIFIED",
      "used_in_response": true
    }
  ],
  "synthesis": {
    "detected": true,
    "reason": "Multiple sensitive sources combined."
  }
```

```
}
```

Synthesis Detection

When synthesis.detected == true, the generated output is a novel artifact -- it did not exist in any source document. This is a critical condition in regulated domains: CUI distribution controls apply to the synthesized output even if no individual source document carried that label. This is the aggregation and compilation threat that traditional IAM cannot prevent; the Threat Model section treats it in full.

10 Identity-Aware Enforcement

ACLX labels encode governance state -- not enforcement decisions. Enforcement decisions are made at the policy engine by evaluating ACLX labels against the identity attributes of the requesting subject. The same ACLX-labeled output may yield different outcomes for different identities.

10.1 Identity Attributes Evaluated

- **role:** Functional role in the requesting system (e.g., Treating Physician, Scheduler, ISSO, Analyst).
- **purpose:** Declared purpose for the request (e.g., treatment, research, audit). Must match an allowed purpose in the SCDO.
- **clearance:** Formal clearance level or authorization tier, if applicable.
- **allowed_distributions:** Set of distribution authority codes the identity holds (e.g., [A, B, D] for CUI).
- **citizenship:** Evaluated against NO_FOREIGN_NATIONALS restrictions.
- **organization:** Evaluated against INTERNAL distribution controls.

10.2 Enforcement Scenario: Same Output, Different Identities

The following example demonstrates differential enforcement for a HIPAA SUPER_PHI labeled response:

Attribute	User A — Scheduler	User B — Treating Physician
Role	Scheduler	Treating Physician
Purpose	Appointment Follow-Up	Treatment
HIPAA Authorized	No	Yes
Enforcement	REDACT	ALLOW

11 Layer 3: Output Classification and Cloud DLP Integration

Layer 3 is where the ACLX label is generated. It is the most novel layer in the framework -- the one that has no direct analog in traditional IAM -- and the one that determines the quality of every downstream enforcement decision. A Layer 4 OPA policy is only as good as the Layer 3 evidence it receives.

The Layer 3 Problem

Section 2.2 established why Layer 3 has no analog in traditional IAM: DLP classifies data that already exists, whereas AI synthesis creates sensitivity at inference time. This section covers one of the detection mechanisms that feeds Layer 3 -- cloud DLP services.

11.1 Detection Mechanisms

Layer 3 uses up to four detection mechanisms in combination. They are ordered by latency and deployment complexity. A production implementation does not require all four from day one -- the phased deployment model in Section 30 describes the sequencing.

Mechanism	How It Works	Strengths	Limitations
Deterministic Rules	Pattern matching on known sensitive forms. Regex for MRN, SSN, DOB; keyword dictionaries for PHI terms, CUI markers, distribution labels.	Sub-millisecond. No external API call required.	Catches known forms only. Cannot detect novel synthesis or conceptual relationships between terms.
Cloud DLP / Sensitive Data Protection	Managed infoType detection via Google Cloud DLP, AWS Macie, or Azure Purview. Classifies PII, PHI, credentials, and custom infoTypes via API.	10-100ms typical. Managed service -- no infrastructure to build.	Returns infoTypes, not ACLX domain/category values. Requires infoType-to-SCDO mapping layer. Data residency constraints for regulated data.
Ontology-Based Reasoning	Structured knowledge model of the data domain. Concept graph with inherited sensitivity. CD4 count is a child of HIV treatment markers, which is PHI not covered by TPO.	1-10ms for graph traversal. Highly precise for known domain concepts.	Requires domain expert build-out. Does not generalize to novel terminology without graph updates.
Semantic / LLM Evaluation	A second, smaller LLM evaluates the output with a structured prompt: given role X and purpose Y, does this response disclose beyond minimum necessary? Returns structured evidence JSON.	Catches novel synthesis that rules and graphs miss. Adapts to novel phrasing.	200-500ms added latency. Reserve for high-risk query patterns only. LLM evaluator itself must be governed.

All four mechanisms produce structured evidence -- a JSON object -- that is consumed by the ACLX label generation engine. The evidence object drives SCDO population: domain, category, subcategory, sensitivity, confidence score, and synthesis flag. The ACLX engine does not receive raw DLP findings; it receives normalized SCDO-compatible evidence.

11.2 Cloud DLP Integration Architecture

Cloud Sensitive Data Protection services (Google Cloud DLP, AWS Macie, Azure Purview) operate as managed infoType detection engines within Layer 3. They do not generate ACLX labels directly -- they generate findings that the ACLX label engine maps to SCDO values. This separation is deliberate: DLP infoType taxonomies and ACLX governance domains are different vocabularies, and the mapping layer must be version-controlled and domain-aware.

Integration Flow

1. The AI model generates a response. The ACLX orchestration layer intercepts the output before delivery.
2. The output text is submitted to the Cloud DLP Inspect API (or equivalent). The request specifies the active infoType set and minimum likelihood threshold.
3. DLP returns a findings array: each finding contains infoType, likelihood score, byte offsets, and quote (if configured).
4. The ACLX infoType mapper translates DLP findings to SCDO domain/category/subcategory values using the mapping table (Section 11.3). Likelihood scores are mapped to confidence.score.
5. Synthesis detection runs in parallel: if multiple infoTypes co-occur in a single response, the synthesis flag is set and the composite sensitivity is elevated.
6. The evidence object is passed to the ACLX label engine, which writes the SCDO and trust blocks.
7. The completed ACLX label is submitted to OPA (Layer 4) for enforcement.

Google Cloud DLP -- Inspect Request Pattern

```
# POST https://dlp.googleapis.com/v2/projects/{project}/content:inspect
{
  "item": {
    "value": "<ai_generated_response_text>"
  },
  "inspectConfig": {
    "infoTypes": [
      { "name": "PERSON_NAME" },
      { "name": "US_SOCIAL_SECURITY_NUMBER" },
      { "name": "MEDICAL_RECORD_NUMBER" },
      { "name": "DATE_OF_BIRTH" },
      { "name": "ICD_CODE" },
      { "name": "US_HEALTHCARE_NPI" },
      { "name": "TECHNICAL_ID" },       // Custom infoType for CUI markers
      { "name": "DISTRIBUTION_LABEL" }  // Custom infoType for DoD dist. labels
    ],
    "minLikelihood": "LIKELY",
    "includeQuote": false,
    "limits": { "maxFindingsPerRequest": 100 }
  }
}

# DLP Response (findings)
{
  "result": {
    "findings": [
      {
        "infoType": { "name": "MEDICAL_RECORD_NUMBER" },
        "likelihood": "VERY_LIKELY",
        "location": { "byteRange": { "start": "42", "end": "52" } }
```

```
      },
      {
        "infoType": { "name": "DATE_OF_BIRTH" },
        "likelihood": "LIKELY",
        "location": { "byteRange": { "start": "98", "end": "108" } }
      }
    ]
  }
}
```

11.3 infoType-to-SCDO Mapping Table

The following table defines the canonical mapping from Cloud DLP infoType findings to ACLX SCDO domain/category/subcategory values. This mapping is maintained as a version-controlled policy artifact alongside the Rego bundle and ACLX schema. Custom infoTypes for domain-specific terms (CUI markers, distribution labels, proprietary identifiers) must be defined in the DLP inspection template and registered in this table.

DLP infoType(s)	Co-occurring infoTypes	Maps to SCDO domain/category/subcategory	Default sensitivity
PERSON_NAME	PHONE_NUMBER EMAIL_ADDRESS	HIPAA / PHI / DEMOGRAPHIC	HIGH
US_SOCIAL_SECURITY_NUMBER	DATE_OF_BIRTH	HIPAA / PHI / SUPER_PHI	HIGH
MEDICAL_RECORD_NUMBER	ICD_CODE	HIPAA / PHI / SUPER_PHI	HIGH
US_HEALTHCARE_NPI	MEDICAL_TERM	HIPAA / PHI / CLINICAL	MODERATE
FINANCIAL_ACCOUNT_NUMBER	CREDIT_CARD_NUMBER	IP / FINANCIAL_DATA / PCI	HIGH
HTTP_COOKIE	OAUTH_CLIENT_SECRET	SAFETY / CREDENTIAL / SECRET	CRITICAL
TECHNICAL_ID (custom)	DISTRIBUTION_LABEL (custom)	CUI / CUI_SP_CTI / [subcategory]	CRITICAL
EU_NATIONAL_ID_NUMBER	EU_PASSPORT	GDPR / PERSONAL_DATA / SPECIAL_CATEGORY	HIGH

Multi-infoType Synthesis Elevation

When DLP findings include co-occurring infoTypes that individually map to different sensitivity levels, the ACLX engine applies synthesis elevation: the composite sensitivity is the maximum of all individual sensitivities, and synthesis.detected is set to true. Example: PERSON_NAME (MODERATE) + MEDICAL_RECORD_NUMBER (HIGH) + ICD_CODE (MODERATE) co-occurring in a single response elevates the composite to HIGH and sets synthesis.detected = true. This is the data aggregation/compilation risk -- no single infoType triggered a critical alert, but the synthesis did.

11.4 DLP-to-SCDO Evidence Object

The ACLX infoType mapper produces a normalized evidence object consumed by the ACLX label engine. This object is not stored externally -- it is an intermediate structure used to populate the SCDO. The mapping from DLP findings to evidence is deterministic and auditable:

```
// Evidence object produced by ACLX infoType mapper
// Input: DLP findings array
// Output: SCDO-compatible evidence for ACLX label engine
{
  "detected_domains": [
    {
      "domain": "HIPAA",
      "category": "PHI",
      "subcategory": "SUPER_PHI",
      "sensitivity": "HIGH",
      "source": "CLOUD_DLP",
      "triggering_infotypes": [ "MEDICAL_RECORD_NUMBER", "DATE_OF_BIRTH" ],
      "confidence": {
        "score": 0.92,           // Derived from DLP likelihood: VERY_LIKELY -> 0.9-1.0
        "level": "HIGH"
      }
    }
  ],
  "synthesis": {
    "detected": true,
    "reason": "Co-occurring PHI infoTypes: MEDICAL_RECORD_NUMBER + DATE_OF_BIRTH",
    "infotype_count": 2
  },
  "dlp_request_id": "dlp-req-a1b2c3",
  "dlp_latency_ms": 47
}
```

11.5 Deidentification: Enforcing REDACT Actions

When OPA returns a REDACT enforcement action, the ACLX enforcement layer must physically redact the sensitive content from the response before delivery. Cloud DLP deidentification APIs provide the mechanism. The deidentification transformation type is determined by the ACLX domain and handling_requirements:

```
# POST https://dlp.googleapis.com/v2/projects/{project}/content:deidentify
{
  "item": { "value": "<response_text_to_redact>" },
  "deidentifyConfig": {
    "infoTypeTransformations": {
      "transformations": [
        {
          "infoTypes": [
            { "name": "MEDICAL_RECORD_NUMBER" },
            { "name": "DATE_OF_BIRTH" }
          ],
          "primitiveTransformation": {
            // HIPAA: replace with [REDACTED] marker
            "replaceWithInfoTypeConfig": {}
            // Alternatives by domain:
            // GDPR pseudonymization: "cryptoDeterministicConfig" (reversible)
            // IP watermark: "dateShiftConfig" or custom surrogate
          }
        }
```

```
      ]
    }
  },
  "inspectConfig": {
    "infoTypes": [
      { "name": "MEDICAL_RECORD_NUMBER" },
      { "name": "DATE_OF_BIRTH" }
    ]
  }
}
```

The deidentified response text is returned to the ACLX enforcement layer and delivered to the client in place of the original. The ACLX audit record captures both the original enforcement action (REDACT) and the deidentification transformation type applied. For BLOCK actions, the deidentify API is not called -- the response is withheld entirely.

11.6 Operational Considerations for Cloud DLP Integration

- **Inspection template versioning:** DLP inspection templates define the active infoType set and likelihood thresholds. Templates must be version-controlled and deployed in coordination with the ACLX infoType mapping table. A template update that adds or removes an infoType changes which SCDO values the ACLX engine can detect.
- **Custom infoTypes for domain-specific terms:** Standard DLP infoType libraries do not cover CUI markers, DoD distribution labels, proprietary algorithm identifiers, or domain-specific clinical terms beyond standard PHI. Custom infoTypes using dictionary lists or regex patterns must be defined for these cases. Custom infoTypes require ongoing maintenance as terminology evolves.
- **Data residency constraints:** Cloud DLP API calls transmit response text to a cloud endpoint. For regulated data (PHI, CUI, ITAR-adjacent), this raises data residency and jurisdictional concerns. Options: (a) use a regionally-scoped DLP endpoint (GCP supports region-specific processing), (b) deploy DLP-equivalent detection on-premises using the same infoType definitions, or (c) restrict Cloud DLP to non-regulated domains and use deterministic rules for CUI/PHI detection.
- **Latency budget:** Cloud DLP inspect calls typically complete in 10-100ms depending on text length and infoType count. This is within the latency budget for synchronous enforcement on most AI interfaces. For streaming LLM responses, DLP inspection should be applied to complete response chunks, not token-by-token streams. The ACLX enforcement layer buffers the complete response before submitting to DLP.
- **Likelihood threshold tuning:** The minLikelihood threshold in the inspection config controls the false positive/false negative tradeoff. LIKELY is the recommended starting threshold. During Phase 1 (log-only), set to POSSIBLE to capture a broader signal for baseline analysis. Tighten to LIKELY or VERY_LIKELY in Phase 2 before enforcement goes live.
- **AWS Macie and Azure Purview equivalence:** AWS Macie and Azure Purview provide equivalent infoType detection capabilities. The ACLX infoType mapper is designed as a translation layer -- the same SCDO evidence object is produced regardless of which cloud DLP service supplies the findings. The mapping table (Section 11.3) must be maintained per-provider as infoType names differ across services.

12 Layer 4: Policy Evaluation Logic

ACLX is designed for evaluation by a policy-as-code engine such as Open Policy Agent (OPA). Policy rules are written in Rego and evaluate ACLX label fields -- populated by Layer 3 detection mechanisms -- against identity

attributes. OPA is Layer 4 in the five-layer framework: it is ready to evaluate output-time policy the moment it receives structured Layer 3 evidence.

12.1 HIPAA SUPER_PHI Rule

```
# BLOCK if SUPER_PHI and purpose is not treatment
deny[msg] {
  domain := input.acl.control_domains[_]
  domain.domain == "HIPAA"
  domain.subcategory == "SUPER_PHI"
  input.identity.purpose != "treatment"
  msg := "SUPER_PHI requires treatment purpose. Request denied."
}
```

12.2 CUI Distribution Rule

```
# BLOCK if CUI Distribution D and identity lacks D clearance
deny[msg] {
  domain := input.acl.control_domains[_]
  domain.domain == "CUI"
  domain.distribution == "D"
  not "D" in input.identity.allowed_distributions
  msg := "CUI Distribution D requires explicit D authorization."
}
```

12.3 Trust-Based Restriction

```
# ESCALATE if trust is LOW and domain is sensitive
escalate[msg] {
  input.acl.trust.level == "LOW"
  domain := input.acl.control_domains[_]
  domain.sensitivity in {"HIGH", "CRITICAL"}
  msg := "Low-trust HIGH/CRITICAL content requires human review before release."
}
```

12.4 Cross-Domain Resolution

```
# Most restrictive action wins across all domains
final_action = "BLOCK"    { deny[_] }
final_action = "ESCALATE" { not deny[_]; escalate[_] }
final_action = "REDACT"   { not deny[_]; not escalate[_]; redact[_] }
final_action = "ALLOW"    { not deny[_]; not escalate[_]; not redact[_] }
```

12.5 OPA Integration

This section describes how the ACLX enforcement layer integrates with Open Policy Agent (OPA) to evaluate ACLX labels at output time. It covers the complete PEP-to-PDP request/response pattern, the OPA input document structure, policy bundle layout, the REST API call pattern, and operational considerations for production deployments.

> **Integration Model**
> The ACLX enforcement layer acts as the Policy Enforcement Point (PEP). OPA acts as the Policy Decision Point (PDP). The ACLX label and the requesting identity attributes are composed into an OPA input document and submitted to OPA via its REST API. OPA evaluates the active Rego policy bundle and returns a decision. The PEP applies the decision to the generated output before delivery.

12.5.1 PEP-to-PDP Request Flow

The following sequence describes the enforcement call at each output boundary event:

8. The ACLX PEP intercepts the generated response before delivery to the requesting client.
9. The PEP composes the OPA input document from (a) the ACLX label attached to the response and (b) the identity attributes of the requesting subject.
10. The PEP submits a POST request to the OPA REST API targeting the acl/enforcement/final_action policy path.
11. OPA evaluates all active Rego rules against the input document and returns a decision object.
12. The PEP reads decision.result and applies the enforcement action: ALLOW, REDACT, BLOCK, or ESCALATE.
13. The PEP writes the decision, input snapshot, and policy version to the immutable audit log.

12.5.2 OPA Input Document Structure

The OPA input document is the composition of the ACLX label and the requesting identity. It is constructed by the ACLX PEP and never modified by the model or the application layer. The structure below is the canonical ACLX input schema for OPA:

```
{
  "input": {

    "acl": {                          // Full ACLX label for the generated response.
      "acl_version": "0.5",
      "content_id": "resp-a1b2c3d4",
      "generated_at": "2026-05-15T14:32:00Z",
      "origin": {
        "system": "aegis-clinical-v2",
        "model": "gpt-4o",
        "session_id": "sess-9f8e7d"
      },
      "trust": {
        "level": "HIGH",
        "justification": "All sources verified via ACLX retrieval."
      },
      "provenance": {
        "mode": "RAG",
        "traceability_score": 0.91,
        "synthesis": { "detected": true, "reason": "Multiple PHI sources combined." }
      },
      "control_domains": [
        {
          "domain": "HIPAA",
          "category": "PHI",
          "subcategory": "SUPER_PHI",
          "sensitivity": "HIGH",
          "distribution": "TREATMENT_ONLY",
          "restrictions": [ "MINIMUM_NECESSARY" ],
          "handling_requirements": [ "REDACTION_REQUIRED" ],
          "confidence": { "score": 0.94, "level": "HIGH" }
        }
      ]
    },
```

```
    "identity": {                    // Requesting subject attributes from IdP/IDP.
      "subject_id": "user-88321",
      "role": "treating_physician",
      "purpose": "treatment",
      "organization": "mercy-general-hospital",
      "citizenship": "US",
      "clearance": null,
      "allowed_distributions": [],
      "attributes": {
        "npi": "1234567890",
        "treating_patient_ids": [ "pt-00412" ]
      }
    },

    "request": {                     // Contextual metadata about the request itself.
      "request_id": "req-zz9921",
      "timestamp": "2026-05-15T14:32:01Z",
      "patient_id": "pt-00412",
      "channel": "clinical-portal"
    }

  }
}
```

12.5.3 OPA REST API Call Pattern

The ACLX PEP submits the input document to OPA using a standard HTTP POST. The policy path acl/enforcement determines which Rego package is evaluated. OPA returns the result of the final_action rule defined in that package.

```
# OPA REST API call from ACLX PEP
POST http://opa-service:8181/v1/data/acl/enforcement
Content-Type: application/json

{ "input": { ... } }         # Full input document as shown in 9.5.2

# OPA response
{
  "result": {
    "final_action": "ALLOW",
    "deny_reasons": [],
    "escalate_reasons": [],
    "redact_reasons": [],
    "policy_version": "acl-policy-v1.6.0",
    "evaluated_at": "2026-05-15T14:32:01.043Z"
  }
}

# BLOCK response example
{
  "result": {
    "final_action": "BLOCK",
    "deny_reasons": [
      "SUPER_PHI requires treatment purpose. Request denied."
    ],
```

```
    "policy_version": "acl-policy-v1.6.0",
    "evaluated_at": "2026-05-15T14:32:01.051Z"
  }
}
```

12.5.4 Complete Policy Bundle Layout

ACLX Rego policies are organized as a bundle loaded into OPA at startup or via bundle API. The canonical bundle structure separates domain rules, cross-domain resolution, helper functions, and the enforcement entrypoint:

```
acl-policy-bundle/
  .manifest                        # Bundle metadata: version, roots
  acl/
    enforcement/
      policy.rego                  # Entrypoint: final_action, deny, escalate, redact
    domains/
      hipaa.rego                   # HIPAA domain rules
      cui.rego                     # CUI/ITAR domain rules
      safety.rego                  # Content safety hard-block rules
      ip.rego                      # Intellectual property rules
      gdpr.rego                    # GDPR domain rules
      export.rego                  # Export control rules
    trust/
      trust_rules.rego             # Trust level evaluation rules
    helpers/
      identity.rego                # Identity attribute helper functions
      distribution.rego            # Distribution authority matching helpers
    data/
      allowed_purposes.json        # Policy data: allowed purpose-by-domain map
      distribution_hierarchy.json  # Distribution authority hierarchy
```

The enforcement/policy.rego entrypoint imports domain rule sets and applies cross-domain resolution. Individual domain files are loaded as a bundle and hot-reloaded without OPA restart when policies change.

12.5.5 Complete Rego Policy -- enforcement/policy.rego

The following is the full content of the enforcement entrypoint. It imports domain rule sets and applies cross-domain resolution to produce a single final_action per evaluation:

```
package acl.enforcement

import rego.v1
import data.acl.domains.hipaa
import data.acl.domains.cui
import data.acl.domains.safety
import data.acl.domains.ip
import data.acl.domains.gdpr
import data.acl.trust.trust_rules

# ── Aggregate deny set across all domain rule sets ──────────────────
deny contains msg if {
  msg := hipaa.deny[_]
}
```

```
deny contains msg if {
  msg := cui.deny[_]
}
deny contains msg if {
  msg := safety.deny[_]
}
deny contains msg if {
  msg := ip.deny[_]
}
deny contains msg if {
  msg := gdpr.deny[_]
}

# ── Aggregate escalate set ──────────────────────────────────────
escalate contains msg if {
  msg := trust_rules.escalate[_]
}
escalate contains msg if {
  some domain in input.acl.control_domains
  domain.review.required == true
  msg := concat("", ["Human review required by authority: ", domain.review.authority])
}

# ── Aggregate redact set ────────────────────────────────────────
redact contains msg if {
  some domain in input.acl.control_domains
  "REDACTION_REQUIRED" in domain.handling_requirements
  not deny[_]
  msg := concat("", ["Redaction required: ", domain.domain])
}

# ── Cross-domain resolution: most restrictive action wins ───────
final_action := "BLOCK"        if { count(deny) > 0 }
final_action := "ESCALATE"     if { count(deny) == 0; count(escalate) > 0 }
final_action := "REDACT"       if { count(deny) == 0; count(escalate) == 0; count(redact)
> 0 }
final_action := "ALLOW"        if { count(deny) == 0; count(escalate) == 0; count(redact)
== 0 }

# ── Decision metadata ───────────────────────────────────────────
deny_reasons    := [msg | msg := deny[_]]
escalate_reasons := [msg | msg := escalate[_]]
redact_reasons  := [msg | msg := redact[_]]
policy_version  := "acl-policy-v1.6.0"
```

12.5.6 Domain Rule File -- acl/domains/hipaa.rego

Each domain file exports a deny set. The enforcement entrypoint aggregates deny sets from all loaded domain files. The following is the HIPAA domain rule file:

```
package acl.domains.hipaa

import rego.v1

# Helper: active HIPAA domains in this response
hipaa_domains := [d | d := input.acl.control_domains[_]; d.domain == "HIPAA"]
```

```
# BLOCK: SUPER_PHI with non-treatment purpose
deny contains msg if {
  some domain in hipaa_domains
  domain.subcategory == "SUPER_PHI"
  input.identity.purpose != "treatment"
  msg := "SUPER_PHI requires treatment purpose."
}

# BLOCK: TREATMENT_ONLY distribution to non-treating identity
deny contains msg if {
  some domain in hipaa_domains
  domain.distribution == "TREATMENT_ONLY"
  not input.identity.role in {"treating_physician", "attending_physician"}
  msg := "TREATMENT_ONLY distribution: identity role does not qualify."
}

# BLOCK: MINIMUM_NECESSARY restriction -- purpose must be in allowed set
deny contains msg if {
  some domain in hipaa_domains
  "MINIMUM_NECESSARY" in domain.restrictions
  allowed := data.acl.data.allowed_purposes.HIPAA
  not input.identity.purpose in allowed
  msg := "MINIMUM_NECESSARY: purpose not in HIPAA allowed purpose set."
}

# BLOCK: PHI to unauthenticated or anonymous identity
deny contains msg if {
  count(hipaa_domains) > 0
  not input.identity.subject_id
  msg := "PHI cannot be released to an unauthenticated identity."
}
```

12.5.7 Policy Data File -- data/allowed_purposes.json

OPA policy data files provide static lookup tables consumed by Rego rules. The allowed_purposes map defines the set of valid declared purposes for each domain. This file is part of the policy bundle and version-controlled alongside Rego rules:

```
{
  "HIPAA": [
    "treatment",
    "payment",
    "healthcare_operations",
    "public_health",
    "research_with_waiver"
  ],
  "CUI": [
    "official_use",
    "authorized_program",
    "need_to_know"
  ],
  "GDPR": [
    "legitimate_interest",
    "contractual_necessity",
    "legal_obligation",
    "vital_interests",
```

```
    "explicit_consent"
  ],
  "IP": [
    "internal_use",
    "authorized_project",
    "legal_review"
  ]
}
```

12.5.8 Operational Considerations

The following considerations apply to production deployments of OPA as the ACLX policy decision point:

- **Latency:** OPA evaluation of ACLX policies typically completes in under 5ms for single-domain inputs. Multi-domain inputs with large control_domains arrays should be benchmarked. For latency-sensitive paths, OPA partial evaluation or compile-time query optimization can reduce per-request overhead.
- **Policy hot-reload:** OPA supports bundle API-based policy updates without process restart. ACLX policy bundles should be versioned and deployed via the OPA bundle API to enable zero-downtime policy changes. The policy_version field in the decision response enables audit correlation to the exact bundle version that made each decision.
- **Input validation:** The ACLX PEP should validate the ACLX label schema before submitting to OPA. Malformed or incomplete input documents should be rejected at the PEP boundary and treated as BLOCK decisions -- never submitted to OPA with missing required fields.
- **Decision logging:** OPA decision logs should be enabled and routed to the same immutable audit store as the ACLX enforcement audit log. OPA's decision log plugin supports structured output compatible with SIEM ingestion. The decision_id in the ACLX audit block should match the OPA decision log entry for end-to-end traceability.
- **High availability:** OPA should be deployed as a sidecar to each ACLX enforcement node rather than as a shared remote service. This eliminates network latency and a single point of failure in the enforcement path. Policy bundles are synchronized to each sidecar instance via the OPA bundle API.
- **Partial evaluation:** For use cases where the identity is known at request time but the ACLX label is not yet generated, OPA partial evaluation can pre-compile identity-specific policy queries. This trades bundle initialization time for reduced per-response evaluation latency at high throughput.

> **AuthZEN Alignment**
> The OPA REST API call pattern described in 9.5.3 is structurally compatible with the OASIS AuthZEN access evaluation API. The input.identity block maps to the AuthZEN subject object and input.request maps to the AuthZEN action/resource objects. Future ACLX versions may define a normative AuthZEN binding to enable PEP/PDP interoperability without OPA-specific dependencies.

13 Multi-Domain Composition

ACLX allows multiple simultaneous governance assertions on a single generated output. This is the normal operating condition for AI systems that retrieve from multiple source systems with different governance regimes.

13.1 Multi-Domain Example

```
"control_domains": [
  {
    "domain": "HIPAA",
    "category": "PHI",
    "subcategory": "SUPER_PHI",
    "sensitivity": "HIGH",
    "distribution": "TREATMENT_ONLY"
  },
  {
    "domain": "EXPORT",
    "category": "ITAR",
    "subcategory": "MISSILE_GUIDANCE",
    "sensitivity": "CRITICAL",
    "distribution": "D",
    "restrictions": [ "NO_FOREIGN_NATIONALS" ]
  },
  {
    "domain": "SAFETY",
    "category": "WEAPONS_INSTRUCTION",
    "sensitivity": "CRITICAL",
    "distribution": "NONE"
  }
]
```

Resolution Outcome

For the example above: SAFETY CRITICAL/NONE triggers BLOCK_REQUIRED as a hard block. Cross-domain resolution returns BLOCK regardless of the requesting identity's HIPAA or CUI authorization. The SAFETY domain is non-negotiable.

14 Reference Enforcement Actions

The following table defines all valid enforcement actions produced by the ACLX evaluation layer. Actions are mutually exclusive at any point in time; cross-domain resolution selects one action per evaluation cycle.

Action	Description	Typical Trigger
ALLOW	Output released to the requesting identity without modification.	Identity holds sufficient clearance and purpose.
REDACT	Sensitive portions removed or masked before delivery.	Identity lacks clearance for specific elements but holds base access.
BLOCK	Entire output denied; only an acknowledgment is returned.	No valid purpose or identity attributes justify release.
ESCALATE	Routed to a human reviewer or privileged decision authority.	Automated policy cannot determine outcome with sufficient confidence.
PARTIAL_ALLOW (reserved)	Subset of output released; remainder redacted or withheld.	Multi-domain output with mixed sensitivity elements.
QUARANTINE (reserved)	Output retained for forensic review; not released or destroyed.	Potential prompt injection or adversarial content detected.

15 ACLX Lifecycle and Enforcement Workflow

ACLX labels are generated, evaluated, and archived in a defined sequence. The following describes the nominal enforcement workflow for a RAG-grounded AI response in a regulated environment.

15.1 Workflow Steps

14. Identity Assertion -- The requesting subject presents identity attributes (role, purpose, clearance) to the ACLX enforcement layer.
15. Retrieval -- The orchestration layer retrieves source documents. Source labels and provenance evidence are captured.
16. Generation -- The AI model generates a response. The orchestration layer injects provenance data and triggers ACLX label generation.
17. Label Assignment -- The ACLX engine assigns domain labels, trust level, and traceability score. The ACLX object is attached to the generated response.
18. Policy Evaluation -- OPA evaluates the ACLX object against the requesting identity. Cross-domain resolution produces a single enforcement action.
19. Enforcement -- The action is applied: response released, redacted, blocked, or escalated.
20. Audit -- The ACLX object, identity snapshot, policy version, and decision are written to an immutable audit log.

16 Content Label — The AI Content Passport

Sections 4 through 14 define the ACLX label: the in-flight governance object that carries domain control assertions through evaluation. The reference implementation wraps that label in a second, outer envelope at release time -- the Content Label, or AI Content Passport. Where the ACLX label is the evaluation input, the Content Label is the signed, portable record of what was decided. It is the artifact that travels with a delivered response, is verified by downstream consumers, and is replayed by auditors.

The Content Label is assembled by the gateway after enforcement resolves. It consolidates data already produced upstream -- the ACLX label, provenance, the policy decision, and the Decision Record -- into one typed, signing-ready envelope. No policy logic lives in the assembler; it is pure consolidation. The current envelope is label_version 1.2.

> **Two objects, one pipeline**
> The ACLX label answers what sensitivity the output carries. The Content Label answers what was decided, by which policy, for which identity, and whether the record can be trusted intact. A consumer that receives only the Content Label can verify and act on a response without re-running classification.

16.1 Content Label Field Reference

The envelope is identity-first: the top-level fields establish provenance for the record artifact itself; nested objects carry the governance state and the decision.

Field	Type	Description

label_version	string	Envelope schema version. Currently 1.2 (1.1 introduced the agent block; 1.2 added chain and memory).
content_id	string	Stable identifier for the governed response. Shared with the audit record.
content_digest	string	sha256 digest of the response text the label binds to. Detects post-issue mutation of the content.
issued_at	ISO-8601	UTC timestamp the label was assembled.
issuer	object	system (default aegis-aclx-gateway), model, and policy_version that produced the decision.
subject	object	Requesting identity: subject, actor_type (human or agent), role, purpose, organization, and an optional agent block (Section 18).
classification	object	domain, sensitivity, and the full control_domains array of evaluated SCDOs.
synthesis	object	Synthesis evidence carried from provenance (detected flag and reasoning).
provenance	object	mode, traceability_score, trust, sources, and source_evidence (Section 22).
handling	object	Resolved action, the union of restrictions and handling_requirements across domains, the strictest review requirement, and the memory directives (Section 21).
chain	array	Verified upstream Content Labels when the response is part of an agent-to-agent chain (Section 19).
decision_record	object	The structured, per-rule decision trace (Section 20).
audit	object	The audit identifiers plus a written flag confirming the event was persisted.
integrity	object	Signature block: alg, key_id, signature, signed_at (Section 17).

16.2 Content Label Template

```
{
  "label_version": "1.2",
  "content_id": "string",
  "content_digest": "sha256:...",
  "issued_at": "ISO-8601",
  "issuer": {
    "system": "aegis-aclx-gateway",
    "model": "string",
    "policy_version": "acl-policy-v1.6.0"
  },
  "subject": {
    "subject": "string", "actor_type": "human | agent",
    "role": "string", "purpose": "string", "organization": "string",
    "agent": { ... }                // present when actor_type == agent
  },
  "classification": {
```

```
    "domain": "string",
    "sensitivity": "LOW | MODERATE | HIGH | CRITICAL",
    "control_domains": [ /* SCDO objects */ ]
  },
  "synthesis": { "detected": false, "reasoning": "string" },
  "provenance": {
    "mode": "RAG | SYNTHETIC", "traceability_score": 0.0,
    "trust": { "level": "...", "justification": "..." },
    "sources": [], "source_evidence": {}
  },
  "handling": {
    "action": "ALLOW | REDACT | BLOCK | ESCALATE",
    "restrictions": [], "handling_requirements": [],
    "review": { "required": false, "authority": "string" },
    "memory": { ... }                // Section 21
  },
  "chain": [],                       // Section 19
  "decision_record": { ... },        // Section 20
  "audit": { "decision_id": "...", "written": true },
  "integrity": {
    "alg": "none | Ed25519", "key_id": null,
    "signature": null, "signed_at": null
  }
}
```

17 Integrity: Tamper-Evident Signing

A governance record is only useful downstream if its integrity can be proven. The Content Label carries an integrity block that binds the entire record -- classification, decision, identity snapshot, and content digest -- under a detached cryptographic signature. A consumer or auditor can verify that the label was issued by a trusted gateway and has not been altered since.

Signing is implemented with Ed25519: deterministic, fast to verify, and producing compact 64-byte signatures. The signature is computed over a canonical JSON serialization of the label (sorted keys, compact separators, UTF-8) with the integrity block removed, so the signature never needs to cover itself.

integrity field	Meaning
alg	none until a key is provisioned; Ed25519 once signing is active.
key_id	Identifier of the public key needed to verify the signature.
signature	Base64-encoded Ed25519 signature over the canonical, integrity-stripped label.
signed_at	UTC timestamp of signing.

17.1 Key Management and Verification Endpoints

The signing key is loaded from configuration as a base64 raw 32-byte seed. When no key is configured the gateway generates an ephemeral key at startup and marks it ephemeral -- signatures remain valid for the process lifetime but will not survive a restart, which is appropriate for development but not for cross-instance trust. The corresponding public keys are published so any party can verify independently.

Keys rotate without breaking in-flight verification. The gateway maintains a keyring -- the current signing key plus any configured prior public keys (SIGNING_PREVIOUS_PUBLIC_KEYS_B64). /trust/keys publishes each key with a status of current or rotated; /verify accepts a label signed by a retired-but-still-published key and returns

verified_by_rotated_key with rotated set to true. This gives operators a grace window to roll keys without invalidating labels signed just before the rollover, and is the mechanism behind the key-staleness guidance for disconnected nodes in the Edge and DDIL section.

Endpoint	Auth	Purpose
GET /trust/keys	None	Publish the gateway public signing keys (key_id to public key) for independent verification.
POST /verify	None	Verify a Content Label signature and return a structured pass or fail reason.

17.2 Verification Outcomes

Verification is total: every label resolves to one of the reasons below. Cross-instance verification is supported by supplying the issuer public key directly, so a label signed by one gateway can be validated by another without shared private state.

Reason	Meaning
ok	Signature valid; the label is intact and issuer-authentic.
verified_by_rotated_key	Valid, but signed by a prior key still published in the keyring during a rotation grace window (rotated = true).
unsigned_or_unsupported_alg	No signature present, or an algorithm the verifier does not support.
unknown_key_id	The key_id is not among the verifier trusted keys.
malformed_signature	The signature field could not be decoded.
signature_mismatch	The signature did not match the canonical label -- the record was altered or signed by a different key.

18 Agent Trust and Attestation

Section 2.2 identified the Layer 1 failure: agentic AI bypasses identity entirely -- no IDP session, no human review, machine-to-machine calls that never trigger authentication. ACLX answers this by making agent posture a first-class enforcement attribute. When the requesting actor_type is agent, the gateway scores the agent and the policy engine gates regulated output on that score.

The model is hybrid, with attestation taking precedence over heuristic scoring. This is the AEGIS attestation integration: a trusted attestor vouches for an agent, and that signed assertion overrides any locally computed guess.

18.1 Trust Bands

Every agent resolves to one of five trust bands. Bands -- not raw scores -- are what the policy rules consume.

Band	Score	Posture
VERIFIED	>= 85	Strongly attested or internally provenanced; eligible for regulated content.
HIGH	>= 70	Trusted; minor risk surface.

MEDIUM	>= 50	Default posture; escalation likely on HIGH/CRITICAL content.
LOW	>= 30	Elevated risk; escalates on sensitive content.
UNTRUSTED	< 30	Denied access to regulated content.

18.2 Attested vs. Heuristic Scoring

Attested path: when an agent presents an unexpired AEGIS attestation, its declared trust_band is authoritative. The attestation carries the attestor identity, an evidence reference, an expiry, and (under the signing model of Section 17) a signature. The label records trust_source = attested and no numeric score.

Heuristic path: with no attestation, the gateway computes a local score from a 50-point base, adjusted by provenance, ownership, and tool surface, then clamped to the 0 to 100 range and mapped to a band. The label records trust_source = heuristic with the numeric trust_score.

Signal	Adjustment	Rationale
Provenance: internal	+30	First-party agent under direct control.
Provenance: managed	+20	Operated under a managed agreement.
Provenance: third_party	-5	Outside the trust boundary.
Provenance: unknown	-15	No provenance asserted.
Named owner present	+10	Accountable party identified.
High-risk tool (each)	-8	external_llm, web_browse, code_exec, network, shell expand the attack surface.
No high-risk tools	+5	Constrained capability.

18.3 Agent-Trust Policy Rules

The policy engine consumes the band and emits structured decisions. These rules run alongside the domain rules and feed the same action resolution (Section 20).

Rule	Effect	Condition
agent_trust.untrusted_regulated	deny	UNTRUSTED agent requesting regulated content.
agent_trust.low_high_sensitivity	escalate	LOW-trust agent on HIGH or CRITICAL content.
agent_trust.unattested_high_sensitivity	escalate	Heuristically-scored (unattested) agent on HIGH or CRITICAL content.
agent_trust.attested_high	pass	Valid AEGIS attestation at VERIFIED or HIGH band.

19 Agent-to-Agent Chain Enforcement

When one agent consumes another agent output, governance must propagate across the hop. ACLX treats a multi-agent workflow as a chain of signed Content Labels. Each downstream gateway verifies the upstream

labels it received, summarizes the chain, and lets the policy engine enforce on the aggregate -- so an upstream block or escalation obligation cannot be laundered by passing content through another agent.

The gateway verifies each upstream label signature (Section 17), then derives a chain summary: how many hops are present, whether any label failed verification, whether anything was blocked or escalated upstream, the maximum sensitivity seen, and the weakest agent trust band in the chain. The verified hops are stored in the Content Label chain array; the summary drives the rules below.

Rule	Effect	Condition
chain.unverified_upstream	deny	An upstream label failed signature verification -- chain integrity is broken.
chain.upstream_blocked	deny	Content was BLOCKED upstream; the block propagates.
chain.upstream_escalated	escalate	Upstream escalation propagates the review obligation downstream.
chain.untrusted_in_chain	escalate	The weakest agent in the chain is UNTRUSTED.

Why signatures matter here

Chain enforcement is only sound if upstream labels cannot be forged. The Ed25519 signing of Section 17 is the foundation: an unverifiable or tampered upstream label is treated as a hard deny rather than trusted on its face.

20 Decision Records and Explainability

Section 4.5 commits ACLX to reproducible, explainable decisions. The implementation realizes this with a structured decision substrate rather than free text. Every policy rule -- domain, trust, agent, chain, and source -- emits a structured entry, and those entries are the authoritative explanation. Nothing downstream is inferred from prose.

20.1 Decision Trace Entry

Each firing rule contributes one entry. The gateway Decision Record Builder sorts entries by effect priority (deny, then escalate, redact, flag, pass) to produce the decision trace.

Field	Type / Enum	Description
rule_id	string	Stable rule identifier, e.g. hipaa.unauthenticated_phi.
effect	enum	pass | flag | redact | escalate | deny.
domain	string	Originating domain or subsystem (hipaa, cui, ip, trust, agent_trust, chain, source, enforcement).
attribute	string	The dotted input path the rule evaluated.
observed	any	The actual value observed at that path.
message	string	Human-readable explanation of the entry.
regulation	string	Regulatory citation backing the rule.
severity	enum	INFO | LOW | MEDIUM | HIGH | CRITICAL.

20.2 Action Resolution Precedence

The final action is resolved from the entry effects by strict precedence. A single deny is decisive; absent denials, escalation wins over redaction, and redaction over allow. This is the canonical resolution implemented in enforcement/policy.rego.

```
deny      present  -> BLOCK
else escalate      -> ESCALATE
else redact        -> REDACT
else               -> ALLOW
```

20.3 Two-Tier Explanation Rendering

Explanations are generated off the enforcement path so they can never alter a decision. Two tiers are provided.

- **Tier 1 (deterministic):** renders the decision record into deterministic, zero-dependency prose -- a fixed headline per action plus the driving rules and the held-versus-required comparison. Always available, no external calls.
- **Tier 2 (grounded narrative):** produces a grounded natural-language explanation via the LLM for end-user, compliance-reviewer, or developer audiences. It is constrained to the decision record and forbidden from inventing rules, regulations, or reasoning. On any error or when disabled it degrades to Tier 1.

> **Egress guardrail**
> Before any decision record is sent to the narrative model, raw evidence quotes are stripped and replaced with a structured-category placeholder unless raw evidence is explicitly permitted. Sensitive matched text does not leave the boundary to produce an explanation of why it was withheld.

Both tiers are exposed through the POST /explain endpoint.

21 Memory Governance Directives

AI systems retain. A response that is safe to display once may be unsafe to store, train on, or carry in long-term agent memory. ACLX derives explicit memory directives from the resolved classification and records them in handling.memory, so downstream agents and stores enforce retention rather than guessing.

Directive	Type	Meaning
can_store	boolean	Ephemeral storage within the trust boundary is permitted.
can_train	boolean	Content may be used for model training.
can_long_term_memory	boolean	Content may persist in agent long-term memory.
can_share	boolean	Content may be shared or redistributed.
retention	enum	none | session_only | short_term | standard.
basis	string[]	Human-readable justification for the directives.

21.1 Derivation Rules

- **BLOCK decision:** no storage, no training, no memory, no sharing; retention = none.
- **No control domains:** all directives permitted; retention = standard.

- **Regulated content:** never train; sharing blocked when HIGH/CRITICAL or when a NO_*_SHARING restriction is present.
- **Retention by sensitivity:** CRITICAL/HIGH -> session_only; MODERATE -> short_term; LOW -> standard.

22 Source-Provenance Inheritance and Elevation

Principle 3.3 makes traceability an enforcement attribute. The implementation goes further: it re-classifies declared sources and reconciles them against the output, so classification cannot be lost or laundered in assembly. The source-evidence analyzer runs before policy evaluation and contributes a source_evidence object to the decision input.

22.1 Inheritance Floor

Output inherits the maximum classification of its sources. Benign-looking text assembled from sensitive sources cannot lower its own sensitivity: when the strongest source outranks the output text classification, the analyzer injects a SOURCE_INHERITED control domain that sets a floor.

22.2 Elevation, Mislabeling, and Spillage

Source rule	Effect	Condition
source.inheritance_floor	flag	Classification inherited from sources -- the floor was applied.
source.classification_elevation	escalate	Output classification exceeds every source -- synthesis created new sensitivity.
source.mislabeled_source	escalate	A source true (re-classified) sensitivity exceeds its declared label.
source.inherited_exceeds_grants	deny	Inherited source distribution exceeds the requester allowed_distributions.

The analyzer ranks sensitivity (LOW, MODERATE, HIGH, CRITICAL) and CUI distribution (A through F) numerically to make these comparisons deterministic, and flags unverified sources so trust scoring can account for them.

23 Groundedness and Hallucination Control

ACLX governs whether output is releasable. Groundedness adds the adjacent question: is the output faithful to its sources. Given the source documents a response was built from, the groundedness layer flags factual claims in the response that are not supported by -- not entailed by -- those sources. Unsupported claims are a quality and risk signal, not a new sensitivity domain, so they flow through the same decision trace as any other finding and escalate for human review. The engine enforces is it grounded the same way it enforces is it sensitive.

> **Sensitivity and faithfulness are different questions**
> A response can be perfectly non-sensitive and still be wrong. Groundedness catches confident, unsupported assertions -- the hallucination failure mode -- that classification alone would pass. It does not block; it routes the response to a human, because a claim that lacks source support needs judgment, not redaction.

23.1 Scope and Method

Groundedness runs only when source text is available -- the retrieval-augmented (RAG) case. With no sources to verify against, faithfulness cannot be assessed here; that gap is already covered by the trust and provenance layer (Section 9), which escalates ungrounded SYNTHETIC, low-traceability output. The two are complementary: provenance governs whether output is traceable at all, groundedness governs whether traceable output actually matches what it cites.

The check is model-assisted. A faithfulness checker is given the SOURCES and the RESPONSE and returns the claims that lack support, each with a one-phrase reason, plus a one-sentence summary. It is deliberately conservative -- it lists only genuinely unsupported claims and ignores wording and style. A verification error (for example, an unreachable model) is surfaced as a non-blocking signal and does not by itself escalate; a confirmed unsupported claim does.

Field	Type	Description
checked	boolean	True when faithfulness was actually evaluated (sources present and model reachable).
supported	boolean	True when no unsupported claims were found.
unsupported_count	integer	Number of unsupported claims found -- drives the ESCALATE rule.
unsupported_claims	array	Up to ten {claim, why} entries naming the unsupported assertions.
reasoning	string	One-sentence summary from the checker.
reason / error	string	Why the check did not run (disabled, or no source text), or a verification error.

23.2 Policy Rules

The summarized result is evaluated by its own policy package and aggregated into the decision trace alongside the domain, trust, agent, chain, and source rules. An unsupported response escalates; a verified-faithful response contributes a pass entry for the audit trail.

Rule	Effect	Condition
groundedness.unsupported_claims	escalate	One or more response claims are not supported by the sources (possible hallucination); human review required.
groundedness.supported	pass	Faithfulness checked and all claims supported -- recorded as trace context.

Unsupported claims carry MEDIUM severity under the AI faithfulness control regulation label, and because the engine resolves the most restrictive effect, a groundedness escalation lifts an otherwise-ALLOW response to ESCALATE. The layer is model-assisted, so at a disconnected or deterministic-only edge node it simply does not assert faithfulness -- and, consistent with the conservative-degradation posture, its absence is never treated as a pass (Section 27). Live flags are observable as the aclx_groundedness_unsupported_total counter.

24 Standards-Corpus Grounding

The detectors and ontologies of Section 11 classify content against built-in knowledge. The standards layer grounds that classification in the protected collection’s own authoritative corpus -- its marking guide or

security classification guide, its controlled vocabulary, and a crosswalk for legacy markings -- plus optional review precedent. It is a retrieval-augmented, citation-bearing signal: every elevation it recommends points back to the governing text that justifies it.

The layer is additive and strictly restriction-only. It never returns an enforcement decision on its own; it produces a recommended minimum action that the gateway merges into the deterministic decision through a most-restrictive merge. With no corpus configured, the decision is exactly what the deterministic engine produced -- the layer can only ever raise restriction, never lower it.

> **Restriction-only by construction**
> The standards recommendation is expressed in the same action space as enforcement and combined by a single max operation: the merged action is the more restrictive of the deterministic action and the standards recommendation. A hard BLOCK is never reduced and an ALLOW floor is never bypassed downward. This monotonic invariant is the whole safety story, and the implementation tests assert it exhaustively.

24.1 Corpus Entry Kinds

A corpus is a set of typed entries. Each entry carries a citation (a human-readable pointer to its governing source), an authoritative excerpt, an optional min_action it asserts, and an optional maps_to classification. Entries are matched by declared terms; matching is deterministic and explainable.

Kind	What it asserts	Example
marking_guide	How the collection’s own marking guide / SCG classifies a topic, with the category it maps to.	Proprietary-under-NDA content is CUI//SP-PROPIN; min_action ESCALATE.
vocabulary	Controlled terms (program nicknames, code words) that are sensitive regardless of context.	Program nickname maps to CUI distribution D.
crosswalk	How to read a legacy or obsolete marking under current policy.	Legacy FOUO maps to CUI Basic per 32 CFR 2002; NOFORN maps to NO_FOREIGN_NATIONALS.
precedent	A restriction learned from a human review verdict (Section 24.3).	A prior DENIED decision raises similar content to ESCALATE.

24.2 Retrieval and Merge

Two complementary lookups run against the response text plus any declared source labels and text. Lexical retrieval scores non-crosswalk entries by how many of their declared terms appear (top matches kept), so a recommendation is always traceable to specific terms. A separate crosswalk scan finds legacy markings by word-boundary match and returns how to re-read them. The strongest min_action across all hits becomes the recommended minimum, and the result -- recommended action, citations, detected legacy markings, and reasons -- is attached to the label under a standards block and carried into provenance. A content-hash snapshot_id identifies exactly which corpus version produced the recommendation, so any audited decision can be reproduced. Standards-driven elevations are counted in the aclx_standards_elevations_total metric.

24.3 Review Precedent

The corpus can be enriched from the human-review feedback loop (Section 25.4) without weakening it. A DENIED verdict -- the reviewer confirms a block or escalation was correct -- becomes a precedent entry that can raise similar future content to ESCALATE. An APPROVED verdict is recorded as informational only (no enforcement weight): precedent may never lower a decision, consistent with the restriction-only invariant. Reloading the corpus refreshes precedent too, closing the feedback-to-enrichment loop at runtime.

24.4 Low-Confidence OCR Escalation

Scanned and legacy documents reach the boundary through OCR (Section 26). Low-confidence OCR must not be silently classified: when the derived text’s OCR confidence falls below the configured threshold, the standards layer raises the recommendation to at least ESCALATE so a human confirms the reading before release. This pairs with document ingestion, where image-only PDFs are flagged for OCR upstream.

24.5 Corpus Administration

The corpus is operated through admin-only endpoints, so a governance owner can update marking guidance without a code change or restart. Because the layer is additive and restriction-only, a reload can never weaken enforcement -- the worst case is an unchanged decision.

Endpoint	Purpose
GET /admin/standards	Runtime corpus status: snapshot_id, entry_count, precedent_entries, and whether precedent is in use.
POST /admin/standards/reload	Hot-reload the corpus and review precedent (and optionally toggle the layer) into the running gateway -- no restart. ADMIN_ROLE only.

25 Platform Services: Gateway, Authorization, and Feedback

The preceding sections describe governance objects and rules. This section documents the runtime that produces them: the gateway API surface, the authorization registry that pre-evaluates legal authorizations, the organization-policy overlay, the feedback loop that tunes decisions over time, and the identity layer.

25.1 Gateway API

Endpoint	Auth	Purpose
POST /evaluate	Yes	Core evaluation of a request and its AI response; returns the decision and Content Label.
POST /generate-and-evaluate	Yes	Generate a response and evaluate it in one call.
POST /explain	Yes	Render a Tier 1 or Tier 2 explanation of a decision (Section 20).
POST /verify	No	Verify a Content Label signature (Section 17).
GET /trust/keys	No	Publish public signing keys.
GET /queue	Yes	List pending and reviewed escalations.
POST /feedback	Yes	Record a human-review verdict.

GET /health	No	Liveness and status.

25.2 Authorization Registry

HIPAA and CUI rules frequently turn on a legal authorization -- a research waiver, a Part 2 consent, a clearance grant. The gateway pre-evaluates the authorization context before policy runs: it filters expired and revoked authorizations and passes only valid ones to the engine, which references the registry result rather than re-deriving validity. Recognized authorization types include the following.

Authorization type	Gates
RESEARCH	research_with_waiver access to PHI.
PART_2_CONSENT / PART_2_COURT_ORDER	Substance Use Disorder records (42 CFR Part 2).
PSYCHOTHERAPY_AUTHORIZATION	Psychotherapy notes (45 CFR 164.508(a)(2)).
HIV_STATE_CONSENT	HIV status disclosure under state law.
CLEARANCE_GRANT	CUI distribution requiring a clearance.

Cross-subject Minimum Necessary (45 CFR 164.514(d)) is enforced at the output layer: the detector checks whether PHI in the response is attributable to subjects outside the authorized-subject scope, using concrete identifiers when DLP is available or MRN/name heuristics otherwise. Out-of-scope PHI escalates; out-of-scope SUPER_PHI is a hard block.

25.3 Organization-Policy Overlay

After OPA resolves a base action, an organization overlay may adjust it within tenant-specific bounds. The overlay is applied in a fixed order so its behavior is predictable.

- **1.** blocked patterns force BLOCK (hard override).
- **2.** a sensitivity threshold can downgrade an ESCALATE when the content sits below the organization escalate-at level.
- **3.** allowed patterns can downgrade ESCALATE to ALLOW.
- **4.** multi-client context escalates or blocks when more than one client identity is present together with PHI.

A failsafe sits above the overlay: if the policy version is unavailable and SUPER_PHI is present, the gateway forces BLOCK rather than failing open.

25.4 Feedback-Driven Confidence

Human-review verdicts are persisted and aggregated into a per-organization confidence score. Over time this tunes borderline decisions: an organization with a consistent permissive history can have an ESCALATE downgraded to ALLOW, while a restrictive history can upgrade an ALLOW to ESCALATE. Verdicts normalize to APPROVED or DENIED, and the queue endpoint surfaces the pending and reviewed items that feed the loop.

25.5 Identity: Multi-Issuer OIDC

Authenticated endpoints validate bearer tokens against a configured set of trusted OIDC issuers (Keycloak, Ping, Okta, Entra, and equivalents). Each issuer is validated by JWKS signature using RS256, ES256, or PS256, with expiry and optional audience checks. Untrusted issuers, failed signatures, expired tokens, and audience mismatches are rejected before evaluation begins.

26 Document Ingestion: Governing Multi-Format Files

The ACLX enforcement core is text-in, decision-out: it deliberately does not parse binary files. Real governance targets, however, arrive as PowerPoint decks, Word documents, Excel workbooks, and PDFs. The extraction adapter is the drop-a-file surface that bridges the two -- it turns a document into text segments and governs each one through the gateway, keeping the enforcement plane format-agnostic. It is a thin service over a reusable extraction library that any in-process consumer can also import directly.

The governing insight is simple: ACLX can only classify what it is given. A naive extract-the-visible-text step silently drops the surfaces where sensitive data actually hides -- and the boundary cannot classify what it cannot see. The adapter pulls those non-rendered surfaces on purpose and tags each segment with where it came from and whether it was hidden.

> **Looks clean on the slide is not actually clean**
> A deck whose visible slide reads Q3 Program Review may carry CUI//SP-CTI, Distribution Statement E, NOFORN in the speaker notes. The rendered slide allows; the hidden note blocks. The same pattern catches PHI in an Excel hidden sheet or a Word tracked-change deletion -- invisible to a human skimming the document, caught at the boundary.

26.1 Hidden Surfaces by Format

For each format the adapter captures the visible content and, critically, the hidden surfaces a visible-text extractor would miss.

Format	Visible	Hidden surfaces captured
PowerPoint (.pptx)	Slide text, tables	Speaker notes, review comments, core metadata.
Word (.docx)	Body, tables, headers/footers	Review comments, tracked-change deletions, core metadata.
Excel (.xlsx)	Visible cells	Hidden / very-hidden sheets, hidden columns, cell comments, defined names, metadata.
PDF (.pdf)	Page text	Document info/metadata; flags image-only/scanned files that need OCR.
Markdown / text	All (split into sections)	--

26.2 Segment Model and Provenance

Extraction produces a list of segments. Each segment carries its text plus provenance: the unit it came from, a human-readable locator, the surface type, and a hidden flag. Hidden surfaces -- notes, comments, tracked-change deletions, metadata, hidden sheets, hidden cells, and defined names -- are exactly the ones a naive

extractor misses. Large surfaces are chunked to stay under the gateway MAX_INPUT_CHARS while preserving fine-grained provenance.

Field	Type	Description
text	string	The extracted text for this segment.
unit	enum	section | slide | sheet | page | document.
locator	string	Human-readable position, e.g. slide 3 notes or sheet Budget hidden columns.
surface	enum	body, table, header, footer, notes, comment, tracked_change, metadata, hidden_sheet, hidden_cells, defined_name.
hidden	boolean	True when the segment did not appear in the normal visible rendering.

The extracted document also reports a byte_sha256 of the original file, a segment count, a hidden-segment count, and any warnings (for example, a PDF with no text layer).

26.3 Document-Level Aggregation

Each segment text is evaluated independently through the gateway with the caller identity and domain. The document decision is the most restrictive segment action, using the same precedence as the enforcement engine: BLOCK over ESCALATE over REDACT over ALLOW. Every non-ALLOW segment is listed in the result, and any hidden segment that drove a restriction is additionally surfaced as a hidden-surface trigger so reviewers see precisely which non-rendered surface caused the outcome.

- **Fail-safe:** a gateway error on a segment fails to ESCALATE for that segment -- never a silent ALLOW.
- **Bounded coverage:** fan-out is capped by MAX_SEGMENTS_EVALUATED, and any truncation is reported as a warning. Coverage is never silently cut.

26.4 Extraction Adapter API

Endpoint	Purpose
GET /health	Status, supported formats, and parser-library availability.
POST /extract	File in, segments out -- preview exactly what ACLX would see, including hidden surfaces.
POST /extract-and-evaluate	File plus identity -- per-segment decisions and the aggregated document decision.

The evaluate form carries domain and subject (required) plus optional role, purpose, actor_type (default human), and organization. Any Authorization header is forwarded to the gateway OIDC gate, so file ingestion inherits the same identity-aware enforcement as direct evaluation.

26.5 Digest Binding and Boundaries

The Content Label content_digest binds to the extracted text the decision was made over. To additionally attest to the original file bytes, carry the adapter byte_sha256 alongside the label. The two together prove both what was decided and which exact file it came from.

- **Out of scope (flagged, not silent):** embedded images, charts, and OLE objects are not classified; image-only or scanned PDFs require OCR upstream -- the adapter flags them rather than passing them silently.
- **Reuse:** the reusable extraction library can be imported directly by any service that prefers in-process extraction over the HTTP adapter.

27 Edge and DDIL Deployment

ACLX is built to run at the edge -- a small office, a forward node, or any Denied, Degraded, Intermittent, Limited (DDIL) environment, including fully disconnected. The design principle is that the enforcement decision is deterministic and local, and the AI-assisted layers are optional and degrade safely. A node never needs the cloud to make a correct, conservative boundary decision.

27.1 Two-Plane Architecture

The three-plane split defined in Section 3.1 is itself the DDIL pattern: governance is authored centrally while enforcement runs autonomously at the edge node, on last-known-good policy, with no call-home on the decision path. The rest of this section details how each component behaves across the connectivity spectrum.

27.2 Component Connectivity

In a fully Denied state the node still authenticates, classifies via the deterministic ontology, runs source-provenance and agent-trust, applies OPA, signs the label, and writes the audit -- with no network. It loses only model-based nuance, and it loses it conservatively.

Component	Needs network	Disconnected behavior
Identity / OIDC validation	JWKS refresh only	Validates against cached JWKS; size the cache TTL to the offline window.
OPA policy engine	No	Enforces on the last-known-good bundle.
Deterministic detectors (ontology, DLP/regex, source-evidence, agent-trust, memory governance)	No	Fully local and deterministic.
Signing (Ed25519)	No	Signs labels with the on-node private key.
Tamper-evident audit	No (to write)	Appends locally; store-and-forward on reconnect.
Label verification	No	Downstream verifies with the cached public key -- no call-back.
Semantic + groundedness (model)	A model	Degrade fail-safe (ESCALATE on regulated content) or use a local model.
Tier-2 narrative	A model	Disabled at the edge; Tier-1 deterministic explanation always renders.

27.3 DDIL by Letter

- **Denied (no link):** gateway and OPA run self-contained. Decision, signing, and audit are fully local. Model-dependent layers fail-safe to ESCALATE or use a local model. No silent pass-through.

- **Degraded (bad link):** disable the Tier-2 narrator, batch and compress audit forwarding, and prefer a local model over a high-latency cloud one to stay inside the timeout budget.
- **Intermittent (comes and goes):** store-and-forward audit replays on reconnect and is centrally verifiable; OPA pulls a fresh bundle on reconnect and runs last-known-good in between; labels verify offline.
- **Limited / low-bandwidth:** content never leaves the node -- only tiny JSON decisions, labels, audit deltas, and policy bundles cross the link, all batchable and compressible. The expensive things (content and model) stay local.

27.4 Conservative Degradation

Two distinct degraded postures are both safe. Deterministic-only is the steady state for a node with no model: the semantic and groundedness layers simply do not run, and the local ontology, OPA, signing, and audit enforce. Configured-but-unreachable is a transient outage: the semantic detector raises on regulated content and fails safe to ESCALATE, while OPA_FAIL_CLOSED makes a missing policy engine BLOCK or ESCALATE regulated content. HIGH, CRITICAL, and SUPER_PHI escalations are never auto-downgraded.

Groundedness (Section 23) is the faithfulness layer in this set: it checks whether the response claims are supported by the cited sources and escalates unsupported (hallucinated) content for review. Like the semantic detector it is model-assisted, so where no model is reachable its absence is treated conservatively -- absence is never a pass.

> **No silent pass-through**
> Whether a node is deterministic-only or temporarily cut off from its model, the boundary holds. The only thing that changes between postures is how much AI nuance the node currently has -- not whether regulated content can slip through unenforced.

27.5 Store-and-Forward Audit

The audit chain is append-only and hash-linked (each event carries the prior hash, forming a chain) and every event is signed. During a blackout the node writes the JSONL locally on a volume; when the link returns it ships the window to the central sink. The central side can then prove the disconnected window is intact: any insertion, deletion, or edit breaks the chain and is reported with the offending sequence number, and the signature check binds the window to the node key.

```
scripts/verify-audit-chain.py audit_events.jsonl --public-key <node-spki-b64>
# OK: 1423 events, chain intact (signatures verified).
```

27.6 Posture and Staleness

A node reports its operational posture at GET /health without leaking secrets -- deterministic-core availability, model provider and configuration, semantic and groundedness availability, mode (full or deterministic-only), fail-closed setting, signing algorithm, audit mode, and deployment profile (cloud, edge, or tactical). Live degradation surfaces in Prometheus counters at /metrics, which carry no PHI.

Counter	Signals
aclx_opa_unavailable_total	Policy engine unreachable -- fail-closed fired.
aclx_semantic_errors_total	Model unreachable on regulated content -- fail-safe ESCALATE fired.

aclx_groundedness_unsupported_total	Unsupported (potentially hallucinated) claims flagged.
aclx_evaluations_total	Throughput.

A disconnected node runs last-known-good policy and trusts cached keys and JWKS, so a staleness policy is required: distribute OPA policy as signed bundles with a max-age after which a node flags rather than silently relaxing; document a manual signing-key rotation and grace procedure and publish prior public keys so labels signed during the window still verify; and size the JWKS cache TTL to the expected offline window, ideally with an in-boundary identity provider.

28 Threat Model

ACLX is designed to address structural threats that arise at the AI output boundary. The following threat categories are in scope for ACLX v0.6.

28.1 Data Aggregation and Compilation

An AI system may synthesize a novel output that combines individually innocuous elements into a sensitive composite. No single source document triggers a control, but the combined output violates distribution authority. ACLX addresses this through synthesis detection and the assignment of composite domain labels.

28.2 Minimum Necessary Violations

An AI system may generate output that exceeds the minimum information necessary for the stated purpose -- even when all source data was legitimately retrieved. ACLX addresses this through purpose-bound distribution controls evaluated at the output boundary.

28.3 Prompt Injection

An adversarial actor may attempt to manipulate the AI model into generating output that bypasses governance controls. ACLX treats injection-suspected output as a BLOCK case: the response is not released and is retained for forensic review. A dedicated QUARANTINE action is reserved for a future engine release.

28.4 Agentic AI Drift

In multi-step agentic workflows, an AI agent may accumulate context across steps, eventually generating output that violates controls that would have been enforced in a single-step request. ACLX requires that labels be re-evaluated at each output step in agentic workflows.

> **Out of Scope -- v0.6**
> Model weight extraction, embedding inversion attacks, and side-channel inference attacks against the underlying model are out of scope for ACLX v0.6. ACLX governs output governance, not model security architecture.

29 Strategic Positioning

ACLX is intended to function as a universal governance abstraction layer for AI-generated content. Rather than embedding enforcement semantics directly into model behavior -- a pattern that is unverifiable, bypassable, and

model-version-dependent -- ACLX externalizes governance into standardized control assertions evaluated by identity-aware policy systems.

29.1 Standards Alignment

- **OPA / Rego:** Native Layer 4 evaluation target. ACLX schema is structured to map directly to OPA input objects. Layer 3 evidence populates the input that OPA has always been ready to evaluate.
- **AuthZEN (OASIS):** ACLX enforcement points are designed to integrate with AuthZEN PEP/PDP interfaces, extending the AuthZEN request model to include AI output objects. AuthZEN currently stops at the request boundary; ACLX defines the output-side extension.
- **SCITT (IETF):** ACLX audit records are candidates for SCITT transparency log entries, enabling third-party verifiability of enforcement decisions at Layer 5.
- **CAEP / SSF:** ACLX enforcement events can be surfaced as Continuous Access Evaluation Protocol signals, enabling downstream identity policy responses to output-time enforcement decisions.
- **SPIFFE/SPIRE:** Workload identity attestation verifies the origin system and model identity in the ACLX origin block -- critical for agentic AI scenarios where no human identity is present.
- **Cloud DLP APIs:** Google Cloud Sensitive Data Protection, AWS Macie, and Azure Purview are Layer 3 detection services. ACLX defines the normalized interface between their infoType findings and the SCDO governance vocabulary that Layer 4 evaluates.

29.2 Architectural Gap

Existing identity standards (OIDC, SCIM, OAuth 2.0) govern authentication and authorization at the request boundary. ACLX addresses the output boundary -- the point where those standards stop. No current standards body owns AI output governance as a defined problem domain. ACLX proposes to fill that gap at the practitioner level while standards processes mature. The five-layer framework, the SCDO schema, the infoType mapping contract, and the OPA evaluation pattern are all practitioner-buildable today with existing infrastructure.

29.3 Why Model Capability Does Not Close the Gap

A reasonable objection to any output-boundary standard: as frontier models grow more capable and better aligned, will the gap ACLX fills simply close on its own? It will not. The gap is structural, not a capability shortfall, and each model generation makes parts of it wider.

- **Separation of duties is not a capability problem:** a regulated entity cannot present model weights to an auditor, and the party being governed cannot be the party attesting to its own compliance. However aligned the model, enforcement must be externalized, reproducible from a signed Decision Record (Section 20), and evaluated by a policy engine the model cannot influence. This is the same reason financial controls do not disappear when employees become more trustworthy.
- **Alignment is not authorization:** model alignment governs what the model will say; ACLX governs who may receive it and under what regulatory handling. A perfectly safe, factually impeccable response describing a patient's psychotherapy notes is still a HIPAA violation when delivered without the required authorization (Section 25.2). Purpose-of-use, clearance, distribution grants, and consent artifacts live in the identity and policy planes -- they are not knowable from the prompt, and a model that guessed at them would itself be a control failure.
- **Synthesis risk scales with capability:** the aggregation and compilation threat (Section 28.1) is a direct function of how well a model synthesizes. A more capable model is better at combining individually releasable sources into a composite that no source system ever labeled -- so the output-classification gap at Layer 3 grows with every model generation rather than shrinking.

- **Capability accelerates the agentic bypass:** more capable models power more autonomous agents, longer machine-to-machine chains, and broader tool use -- exactly the actors that bypass session-based identity (Section 2.2, Layer 1). Agent trust, attestation, and chain enforcement (Sections 18 and 19) become more load-bearing as capability increases, not less.

What better models do change is the economics, and the architecture is built to absorb that. The model-assisted layers -- semantic detection, groundedness, narrative explanation -- are swappable components that improve with each generation, reducing false escalations through the feedback loop (Section 25.4) while the deterministic invariants hold. Detection quality is expected to commoditize; the durable contribution of ACLX is the enforcement contract around it: the label schema, identity-aware evaluation, the signed portable passport, and the reproducible audit trail. A better classifier makes that contract cheaper to honor. It does not replace it.

> **The test that survives every model generation**
> Ask not "can the model classify this correctly?" but "can this decision be proven, to a party that does not trust the model, to have been made under the right policy for the right identity?" No improvement in model capability answers the second question. ACLX exists to answer it.

30 Phased Deployment: Log First, Enforce Second

ACLX enforcement is not a day-one switch. The phased deployment model below reflects the operational reality that false positive rates must be measured before enforcement goes live. Log-first, enforce-second is not a concession -- it is the correct sequencing for any output governance system where the cost of a false positive (blocking legitimate clinical information) may be as consequential as the cost of a false negative.

Phase	Governing Question	Actions	ACLX / DLP Configuration
Phase 1 Days 1-90	What is my AI actually producing?	Log response streams. Do not intercept. Identify cross-source synthesis queries. Measure outputs exceeding thresholds. Build normal-behavior baseline. Quantify false positive rate.	DLP Inspect API in log-only mode. Deterministic rules applied to logs offline. No OPA enforcement -- Layer 4 is passive.
Phase 2 Days 91-180	Where is my actual risk concentrated?	Evaluate highest-risk query patterns only. Require human review for elevation events. Build override workflow with logging. Refine detection against Phase 1 baseline.	DLP + deterministic rules feeding OPA in shadow mode. ESCALATE actions routed to human review. BLOCK enforcement enabled for CRITICAL/NONE patterns only.
Phase 3 Day 181+	Does my governance posture match my AI footprint?	Extend evaluation across all patterns. Integrate into SIEM and compliance reporting. Add agentic AI to evaluation scope. Automate regulatory reporting from logs.	Full five-layer enforcement active. Semantic LLM evaluation enabled for high-risk patterns. IGA integration for AI output audit trail. SCITT transparency log enrollment.

The Governing Question at Each Phase

Phase 1: What is my AI actually producing? Phase 2: Where is my actual risk concentrated? Phase 3: Does my governance posture match my AI footprint? Do not move to Phase 2 until Phase 1 has produced a documented baseline. Do not move to Phase 3 until Phase 2 enforcement has been stable for 30 days.

30.1 Three Diagnostic Questions

Before beginning Phase 1, answer these three questions to scope your initial deployment:

- **Understand your exposure:** List every AI system touching regulated data. For each: what sources does it query across simultaneously? Request three sample outputs and evaluate them against your minimum necessary criteria. If you cannot reconstruct what the model combined from source data, you have a gap. That gap is your starting point.
- **Test before you enforce:** Run your riskiest query pattern in logging mode for 30 days. Apply your minimum necessary criteria to the outputs manually. Count how many would trigger a violation under existing policy. That number is your risk baseline and your funding justification. Manual review violations equal automate detection.
- **Wire into what you have:** Check whether your policy engine accepts new evidence types -- most can. Confirm your SIEM ingests AI decision logs -- it almost certainly can. Identify who owns output-time policy in your organization. If nobody does, that ownership gap is the finding. You are not building new infrastructure. You are extending what already exists.

32 Summary Statement

The Access Control Label eXtended normalizes heterogeneous regulatory, security, safety, and legal control regimes into standardized machine-readable control assertions capable of identity-aware policy evaluation at the AI output boundary.

ACLX is Layer 3 of a five-layer adaptive access control framework. It generates the structured evidence that Layer 4 policy engines (OPA, Cedar, AuthZEN) evaluate, and it feeds the output-time decisions that Layer 5 audit and governance systems need to close their AI compliance gap. Cloud DLP services (Google Cloud Sensitive Data Protection, AWS Macie, Azure Purview) are the managed infoType detection engines that feed Layer 3 -- they detect what is present in the AI response, and the ACLX infoType mapper translates their findings into governance-ready SCDO values.

ACLX does not replace existing IAM infrastructure. It extends that infrastructure to the point where it currently stops: the moment an AI system generates a response. Layers 1, 2, 4, and 5 already exist in most environments. The only missing layer is Layer 3 -- output classification. ACLX is the specification that makes Layer 3 buildable and interoperable.

The Core Assertion

Traditional IAM enforces who can ask the question. ACLX enforces who can receive the answer -- and how much of it. Cloud DLP detects what the answer contains. OPA decides what to do about it. The five-layer framework connects all three.

A Version History

Version	Date	Author	Changes
0.1	2026-01	C. Hunt	Initial draft. Core schema and HIPAA example.
0.2	2026-03	C. Hunt	Added CUI/ITAR domain. Trust model introduced.
0.3	2026-05	C. Hunt	Multi-domain support. OPA evaluation pattern. Provenance traceability model.
0.4	2026-05	C. Hunt	Five-Layer Framework. Cloud DLP integration (GCP, AWS, Azure). infoType-to-SCDO mapping. Deidentification pattern. Phased deployment model. Regulatory regime table. Layer 3 detection mechanisms.
0.5	2026-06	C. Hunt	Content Label (AI Content Passport) envelope and Ed25519 tamper-evident signing. Agent trust and AEGIS attestation. Agent-to-agent chain enforcement. Structured Decision Records and two-tier explainability. Memory governance directives. Source-provenance inheritance and elevation. Platform services: gateway API, authorization registry, org-policy overlay, feedback loop, multi-issuer OIDC. Multi-format document ingestion (PowerPoint, Word, Excel, PDF) with hidden-surface capture. Standards-corpus grounding (marking guide / vocabulary / legacy-marking crosswalk / review precedent; restriction-only) with hot-reload admin. Signing key-rotation grace (keyring + prior-key verification). Groundedness / hallucination control (faithfulness checking of RAG output). Edge / DDIL deployment profile (deterministic local core, fail-closed/fail-safe degradation, store-and-forward hash-chained audit). Explicit positioning of ACLX within the AEGIS framework (three-plane governance / enforcement / delivery model). ACLX expands to Access Control Label eXtended (name decoupled from the AEGIS framework); ACLX trademark noted. Reconciled schema to implementation (acl_version 0.5; RAG/SYNTHETIC provenance). Planned next: SCITT transparency log, CAEP event bindings, SPIFFE workload identity, AuthZEN normative binding.
0.6	2026-07	Chris Hunt	Added Section 31 Knowledge Integrity: Trusted Knowledge Zones (graded, zone-based source trust), per-claim assertion records with authoritative-corpus corroboration (Ed25519-signed, versioned; poisoned-RAG defense), model-as-source trust with offline drift detection, and multi-model consensus as a secondary flag-only anomaly signal. All four are additive, feature-flagged (default off), and cap at ESCALATE. Document status advanced to v0.6.