Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Designing a Governed Lakehouse for Distributed Cloud Data and AI Systems: Practical Reference Architecture, Rollout Plan, and Trade-offs

Ankit Joshi
Independent Researcher, USA

ARTICLE INFO

Received: 05 March 2026
Revised: 09 April 2026
Accepted:22 April 2026

ABSTRACT

Enterprise lakehouse initiatives can stall when governance is treated as tool configuration and informal process rather than as a first-class platform capability. Modern lakehouse deployments introduce multiple enforcement surfaces (batch engines, interactive SQL, gateways, services) and typically require high availability within a region (for example, across availability zones) [1]. In some organizations, deployments also span regions for disaster recovery (DR) or cross-region access—creating governance-state drift, inconsistent policy interpretation, and expensive manual audit work if control-plane semantics are not explicit. This paper presents a governance-first reference architecture for distributed lakehouse environments, emphasizing separation of control-plane responsibilities from data-plane execution. It provides a practical blueprint for (i) authoritative dataset identity and lifecycle management, (ii) policy specification, versioning, and portability across engines, (iii) schema evolution and contract-based change management, (iv) audit-grade evidence capture and provenance, and (v) governance-state distribution semantics for multi-instance, multi-AZ, and DR scenarios. The article also includes operational acceptance checks and an adoption roadmap to guide phased implementation and prevent governance debt as platform adoption and AI usage expand [8].

Keywords: Data Governance, Lakehouse Architecture, Distributed Systems, Control Plane, Schema Evolution, Auditability, High Availability, Disaster Recovery.

I. INTRODUCTION

1.1 The Enterprise Data Platform Evolution

Enterprises are moving from siloed warehouses toward integrated lakehouse architectures to support BI, streaming analytics, and ML/AI workloads on shared data foundations [1]. This consolidation increases reuse and agility, but it also expands governance scope across teams, tools, and execution environments.

1.2 The Governance Gap in Lakehouse Implementations

In practice, lakehouse programs often hit governance limits that pure performance scaling cannot solve: fragmented dataset identity across catalogs, permission drift through ad hoc exceptions, and downstream breakage when producers change schemas without coordinated safeguards. These issues create hidden operational costs: incident response, emergency access changes, pipeline restoration, and manual compliance preparation.

In many environments, the core issue is not a missing tool; it is missing control-plane semantics: a reliable way to define datasets, express policies, distribute governance state, and capture evidence consistently across a distributed enforcement surface.

1.3 Problem Statement and Scope

This paper addresses governance scalability by treating governance as engineered infrastructure rather than a reactive compliance overlay. It focuses on implementation-agnostic principles applicable across heterogeneous enterprise stacks. The scope includes deployments that require high availability (for example, multi-AZ) and, in

some organizations, cross-region DR or multi-region access, where geographic separation introduces explicit replication and degraded-operation requirements.

1.4 Article Structure and Approach

The article presents a governed lakehouse reference architecture and a set of implementation capabilities that can be adopted incrementally: control-plane/data-plane separation, versioned governance artifacts, effective-state observability, schema contracts, evidence capture, distributed governance-state semantics (multi-instance/multi-AZ/DR), and AI readiness controls.

1.5 Methodology and Basis of Observations

This guidance synthesizes practitioner observations from enterprise governance implementations over a multi-year period. The analysis draws on anonymized post-incident reviews, compliance and audit findings, platform migration retrospectives, and structured interviews with platform engineering teams. Identifying details have been removed. The goal is not statistical prevalence estimation, but to extract operationally useful design decisions and trade-offs that recur across real deployments.

1.6 Unique Contribution and Scope Definition

This paper contributes: (1) a reference architecture for governance in distributed lakehouses, (2) explicit artifact lifecycle and distribution semantics, (3) verification/operability signals to validate correctness in production, and (4) a phased adoption roadmap and decision matrix to sequence implementation and reduce governance debt.

II. FOUNDATIONAL GOVERNANCE ARCHITECTURE

2.1 Control Plane versus Data Plane Separation

A governed lakehouse separates data plane responsibilities (storage, compute, transactions, query execution) from control plane responsibilities (dataset identity, policy semantics, distribution, evidence capture, audit retention, and correctness verification). This separation helps prevent governance logic from being embedded inside a single engine and enables consistent semantics as new access paths and engines are added. The governance failures that arise when this separation is absent — enforcement drift across engines, governance-state propagation gaps, and permission inconsistency — are documented recurring patterns in enterprise deployments; this paper specifies what the control plane must contain and how to build and operate it.

A practical implementation defines a semantic policy core and uses adapters/compilers to translate the core into enforcement-native mechanisms across engines, gateways, and services.

2.2 Governance Artifacts as Versioned, Distributed Objects

Governance must be expressed as explicit artifacts that can be reviewed, versioned, promoted, rolled back, and audited—rather than as scattered UI settings across tools. Artifact-based governance provides the technical precondition for controlled rollout and skew measurement. Whether skew remains bounded in practice depends on the implementation of the distribution, activation event, and conformance monitoring components described in Section 2.5. Without those components, versioned artifacts alone do not guarantee bounded skew.

2.2.1 What counts as a “governance artifact”

The following artifact types are introduced here for orientation. Their role in preventing governance failures — including metadata fragmentation, enforcement drift, and lineage gaps — is grounded in documented enterprise failure patterns; this paper's focus is artifact packaging, lifecycle, and distribution.

At minimum, a governed lakehouse should externalize these artifact types:

- **Dataset registration artifacts:** dataset ID, canonical name, owners/stewards, tier/SLO class, classifications, lifecycle state (draft/published/deprecated/retired), and linkage to physical locations.
- **Policy artifacts:** access rules (RBAC/ABAC), row/column rules, masking/obligations, deny precedence, break-glass rules (with expiry), and enforcement scope.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Schema contract artifacts:** schema definition + compatibility rules (allowed changes), deprecation windows, and validation checks.
- **Evidence artifacts:** decision-log schema, required fields, retention rules, and audit-store access controls.
- **Lineage/provenance artifacts** (optional but recommended): capture boundaries, required provenance fields, and retention/immutability rules.

2.2.2 Artifact packaging and identity

Each artifact should have:

• Stable identity (e.g., dataset-id + artifact-type)

• Version (semantic or monotonic) and immutability once published

• Provenance: who authored/approved it, when, and from which source (PR/commit)

• Integrity: checksums for all artifacts; signing is required for tier-1 / regulated scopes and recommended otherwise, so tampering is detectable and activation can be blocked.

A practical approach is to package artifacts into bundles aligned to rollout units (e.g., "policy bundle for domain X" or "release bundle for dataset Y"), where a bundle contains the policy + contract + evidence config required for that scope.

2.2.3 Lifecycle: author → validate → version → distribute → enforce → audit

A reliable lifecycle turns governance into an operational pipeline:

1. **Author:** changes are proposed as text (policy-as-code / config-as-code) with reviewers.
2. **Validate:** static checks (syntax), semantic checks (deny precedence, conflict detection), and conformance checks against supported enforcement capabilities.
3. **Version & freeze:** publish an immutable artifact version; record approvals and rationale.
4. **Distribute:** push artifacts to enforcement points via a controlled channel (pull-based preferred for reliability).
5. Enforce: enforcement points activate the new version with explicit activation semantics.
6. Audit: every access decision links to an artifact version; audits can reproduce "effective state" at any time.

2.2.4 Rollout, rollback, and compatibility

Artifact versioning is only useful if rollout is controlled:

• Progressive rollout: canary a subset of enforcement points/regions before global activation.

• Rollback: ability to revert to a known-good artifact version quickly.

• Compatibility contracts: enforcement points should declare supported policy/features; the pipeline should block or require a safe fallback (e.g., deny / gateway enforcement) for artifacts that cannot be enforced consistently.

Gate override protocol. When an operator needs to override a blocked rollout gate:

- The operator submits justification via the rollout controller (free text, minimum 50 characters).
- A named approver from the data platform lead tier confirms the override.
- Controller creates a time-bounded exception (maximum 24 hours; auto-expires).
- An immutable audit log entry is created: {bundle_version, gate_type, justification, approved_by, expiry_ts, created_ts}.
- At expiry, the gate is automatically re-enforced. If the issue is not resolved, the rollout halts again.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2.2.5 Operational acceptance checks

- % of governance state represented as versioned artifacts (vs ad hoc settings)
- Median/P95 time to propagate a new artifact version to all enforcement points
- Rollback time and success rate
- Rate of "non-enforceable" policy changes caught by validation (healthy friction)

2.3 Metadata Authority and Dataset Identity

To prevent identity drift across catalogs and access paths, a governed lakehouse should maintain an authoritative dataset registry (aligned with FAIR principles F1 (globally unique, persistent identifiers) and F2 (rich metadata associated with the identifier) [3], adapted here for operational enterprise use rather than scientific data publication) that records:

- Unique dataset ID and canonical name
- Owner/steward and escalation path
- Sensitivity/classification
- Lifecycle state (draft/published/deprecated/retired)
- Contract baseline and tier (e.g., tier-1 vs best-effort)

Publishing gates ensure datasets are not treated as "enterprise assets" until identity, ownership, and baseline governance metadata exist.

The operational failure this registry prevents — multiple teams using incompatible dataset definitions, ownership unclear during incidents, classification inconsistent across engines — is a well-documented enterprise pattern. The minimum registry fields required to address it are specified through the Phase 0 exit criteria in Section 3.5.

2.4 Hierarchical Access Control Framework

A consistent access control model should support inheritance and delegation while remaining deny-by-default [4]. Inheritance applies only within explicitly granted scopes, and exceptions must be recorded with reason, expiry, and audit evidence.

2.4.1 Credential lifecycle on tightening events

When a revocation or classification upgrade occurs, the control plane MUST:

1. Publish a new bundle with a revocation flag and a short TTL override for the affected datasets.
2. Notify token issuers to reduce remaining token TTL to max(0, original_TTL – time_since_issue).
3. Enforcement surfaces MUST re-validate credentials on the next request if policy_bundle_version_at_credential_issue < current_activated_bundle_version.

In-flight requests initiated before the tightening event MUST be completed against the decision-used state locked at request initiation. A surface MUST NOT change its decision-used state mid-request; cache updates take effect only on subsequent requests.

2.5 Effective Governance State Observability

Distributed governance can fail quietly when operators cannot observe "what is currently true" at the enforcement edge. Observability is not a dashboard add-on; it is part of the governance contract. This paper recommends that every enforcement surface expose its effective governance state in a queryable, measurable form. The interface contract this paper proposes is specified in Table 1.

Definition (effective governance state). The *effective governance state* of a dataset is the combination of (a) the currently activated governance artifacts (policy bundle version, classification snapshot, schema contract

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

version) and (b) the enforcement metadata actually used at decision time (cache version, decision timestamp, evaluation mode, and applicable overrides). "Effective" is therefore measured at *enforcement time*, not at authoring time.

2.5.1 What "effective state" means (make it decision-reproducible)

Define **two states**, and require surfaces to report both:

- **Cached state**: the newest verified governance artifacts currently stored locally on the surface (policy bundle, dataset registry snapshot, contract/obligation config).
- **Decision-used state**: the governance artifact versions actually used to evaluate an access decision at time *t*.

For each enforcement surface (engine, gateway, service), effective state MUST include:

- **Policy bundle version** (plus activated timestamp, and previous version)
- **Dataset registry snapshot/version** used for decisions (explicit snapshot ID, not "latest")
- **Contract/obligation configuration hash** (masking rules, row filters, purpose constraints)
- **Capability set** of the surface (features supported; used for rollout gating)
- **Degraded-mode flags** (whether the surface is enforcing under partition/failover rules)
- **Region/zone context** and dependency health (control-plane reachability, cache freshness)

Normative requirement: Every access decision log MUST include a **correlation ID** that links the decision to the **decision-used state** (policy version + registry snapshot + obligation hash). Without this linkage, audits cannot reproduce effective state.

2.5.2 Required observability signals (SLO-grade)

At minimum, expose these signals per surface and per region:

- **Version skew**: max/min policy version across surfaces; count of out-of-date surfaces
- **Propagation lag**: time from publish → activated everywhere (tracked separately for revocations/classification tightening vs grants/relaxations)
- **Decision traceability completeness**: % of decisions that include (policy version + dataset ID + evaluated attributes + obligation outcomes)
- **Drift/non-conformance**: surfaces whose decisions diverge from conformance tests on a shared test vector suite
- **Degraded-mode indicators**: number of surfaces in partition/failover rules; duration; affected scopes
- **Rollback activity**: rollback frequency, rollback success rate, and post-rollback conformance status

2.5.3 Implementation contract (endpoints + events)

To avoid "observability by convention", define a minimal contract:

- **State query endpoint** (or equivalent telemetry):
  GET /governance_state returns cached state + decision-used state metadata, capability set, and degraded-mode flags.
- **Activation event**: each surface emits an event when it activates a new policy bundle, including (bundle version, timestamp, result, reason).

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Central aggregator**: a service aggregates per-surface state and produces skew/lag distributions and "out-of-policy" alarms.

Table 1 spells out the minimum interface contract required to make effective-state observability testable and enforceable across surfaces.

Interface / signal	Required fields / behavior	Produced by	Consumed by	Failure if missing
GET /governance_state	policy_bundle_version; activated_ts; registry_snapshot_id; obligation_hash; capability_set; degraded_mode_flags; freshness/TTL metadata	Every enforcement surface	Central aggregator; operators	Audits cannot reproduce "what was true"; drift can't be measured
Activation event	bundle_version; timestamp; result; reason; surface_id; region	Every enforcement surface	Central aggregator; rollout controller	Rollout cannot prove activation; skew/lag data unreliable
Skew/lag aggregation	max/min versions; out-of-date count; propagation lag distribution; revocations tracked separately	Central aggregator	Oncall + SRE dashboards; rollout gates	No SLO enforcement; partial rollout drift undetected
Decision log correlation	correlation_id linking decision to decision-used state; dataset_id; policy_version; obligation outcomes	Enforcement logging pipeline	Audit + incident response	Evidence chain breaks; "decision explainability" becomes narrative only

Table 1. Minimal effective-state observability contract (interfaces).

obligation_hash definition: The obligation_hash field is the SHA-256 of the canonical JSON serialization of {masking_rules, row_filters, purpose_constraints} drawn from the activated bundle, with keys sorted alphabetically and no whitespace. This allows any consumer of the /governance_state response to verify that the obligation configuration in effect matches an expected value without transmitting the full config in every response.

Effective-state locking semantics: A surface maintains two states: the cached state (updated from the distribution channel on each poll or push event) and the decision-used state (the state active when a given request was initiated). These may differ during a cache update. A surface MUST lock the decision-used state at request initiation and MUST NOT change it mid-request. Decision log entries MUST reference the decision-used state, not the cached state at time of logging.

2.5.4 Cache, TTL, and failure semantics

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

This section defines the safety semantics that make pull-based distribution defensible under partitions and control-plane outages. A pull-based distribution model with local cache is viable only if its failure behavior is explicit:

- **Bounded staleness:** Define a maximum allowed propagation lag per tier and operation class. The following are reference values — tune to your organization's risk posture: [6] (bounded staleness concept; the implementation semantics in this architecture are deterministic TTL-based rather than probabilistic quorum-based)

Tier	Operation	P95 Propagation Target
Tier-1	Revocation / tightening	≤ 5 minutes
Tier-1	Grant / loosening	≤ 15 minutes
Best-effort	Any	≤ 60 minutes

  Revocations and grants MUST be tracked separately in the central aggregator — a slow revocation is a security event; a slow grant is an availability inconvenience.

  When TTL expires and the control plane is unreachable: Sensitive scope means datasets with tier set to tier-1 in the registry, evaluated from the surface's last-known-good registry snapshot. For tier-1 datasets, deny-by-default applies on TTL expiry. For best-effort datasets, last-known-good state persists until the control plane is reachable.

- **Integrity:** tier-1 / regulatory scope bundles MUST be signed (and signature verification failure MUST block activation). For lower tiers, signing is recommended; at minimum, enforce checksums + immutable storage + provenance.
- **Control-plane unreachable:** specify degraded behavior explicitly (e.g., continue serving using last-known-good within TTL for non-sensitive scopes; deny-by-default for sensitive scopes).

- **Revocation fast path:** Push-on-revoke and tighter polling as a safety net — these are complementary, not alternatives. Use push-on-revoke for surfaces with confirmed control plane connectivity; use tighter polling as the fallback for surfaces behind network boundaries or where push delivery cannot be confirmed. If push-on-revoke delivery is not confirmed within 30 seconds, the rollout controller MUST trigger an immediate poll for all surfaces covering the affected dataset.
  The distribution channel MUST use monotonic bundle versioning. Surfaces MUST NOT activate a lower-version bundle after activating a higher-version bundle, regardless of which arrives first.

2.5.5 Rollout, rollback, and safety constraints

Observability must support control actions:

- **Progressive rollout guardrails:** block broad rollout if canary surfaces fail conformance or exceed skew/lag thresholds.
- **Rollback readiness:** surfaces must revert to prior bundles quickly; record attempts and outcomes.
- **Rollback safety rule:** "roll back" must not silently re-enable revoked access. Revocations require explicit break-glass justification, expiry, and audit evidence.

2.5.6 Rollback safety invariant

A rollback to a prior bundle version MUST NOT silently re-enable a previously revoked permission. The rollout controller enforces this as follows:

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

1. The controller maintains a separate revocation log — append-only, monotonically versioned, stored independently of bundle versions.
2. During rollback from bundle N to bundle N-1, the controller computes: effective_policy = bundle_N-1 UNION revocation_log_since_N-1.
3. Surfaces that cannot apply the revocation overlay MUST route access to affected datasets through an enforcement gateway that can.
4. Any rollback that would re-enable a previously revoked permission requires break-glass authorization: written justification, explicit approver identity, time-bounded exception window (maximum 24 hours), and an immutable audit log entry. The rollout controller MUST block automatic rollback in this case.

2.5.7 Operational acceptance checks (examples)

- Policy version skew window (max minutes/hours allowed by tier)
- P95 propagation lag (separately for revocations vs grants)
- % of decisions with complete trace context (policy version + dataset ID + obligation outcomes)
- Drift rate: number of non-conformant surfaces per week
- Mean time to detect (MTTD) and mean time to remediate (MTTR) governance-state drift

III. OPERATIONAL GOVERNANCE MODULES

3.1 Schema Contract Management

Schema evolution must be governed by explicit contracts rather than informal coordination. A practical design includes:

- Contract per dataset: compatibility rules and allowed changes
- Pre-merge gates for changes (CI checks)
- Migration windows (dual write/read) when needed
- Rollback protocol that preserves auditability

3.1.1 Operational Acceptance Checks

downstream breakage rate attributable to schema changes; time-to-detect breaking changes; % of tier-1 datasets with enforced compatibility gates. Table 2 summarizes a practical change-class framework that teams can use to define contract rules, CI gates, and rollback posture.

Change class	Examples	Default compatibility stance	Required workflow (minimum)	Validation / operational acceptance checks	Rollback posture
Additive (low risk)	Add new nullable column; add new optional field; add new partition value	Backward compatible if consumers tolerate unknown fields / nulls	CI schema diff + contract rule check; publish contract version; notify subscribers	% tier-1 datasets with CI contract gate; downstream breakage attributable to	Prefer forward-only; rollback rarely needed

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

				schema change	
Compatible transform	Type widening (int→long); enum extension where "unknown" is handled	Conditionally compatible (must be explicitly allowed in contract)	CI gate + consumer capability declaration; staged rollout (canary consumers)	Canary consumer pass rate; "unsupported change" rejections (healthy friction)	Rollback allowed if auditability preserved
Breaking (high risk)	Rename/remove field; change semantic meaning; make nullable→non-null; change key structure	Not compatible unless migration path exists	Migration plan required (dual write/read window); deprecation period; explicit approval	Breakage detection time; % changes with migration window; incidents from schema changes	Rollback must preserve evidence + reproducibility
Semantic (sneaky risk)	Same schema shape but meaning changes (units, currency, timezone)	Assume incompatible unless versioned semantics declared	Requires contract note + validation tests; explicit "semantic version bump"	Failed semantic tests; post-change anomaly alerts; consumer acknowledgements	Rollback strongly preferred if detected late

Table 2. Schema contract change classes and required controls.

Use these change classes to keep "contract enforcement" concrete (what is allowed, what is blocked, what requires migration), rather than aspirational.

3.2 Policy Control Plane & Multi-Engine Portability

As organizations add engines (batch, interactive SQL, streaming), gateways, and services, governance fails if "policy meaning" varies by access path. A policy control plane must define a semantic core and provide controlled translation to each enforcement surface.

3.2.1 Define the semantic policy core

The core should specify, in a technology-neutral way:

- Subjects (user/service identity, groups, roles)
- Objects (datasets, columns, partitions, models/features)
- Actions (read/write/admin; in AI-heavy environments may also include export/use-for-training)

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

• Attributes (classification, region, time; some organizations also use purpose/device posture via identity or conditional-access layers)

• Obligations (masking, filtering, watermarking, logging, approval)

• Precedence rules (deny overrides allow; break-glass with expiry)

This core is the single source of truth. Engine-specific representations are derived outputs, not the authoring surface.

3.2.2 Build adapters/compilers per enforcement surface

Each surface implements a compiler that maps the semantic core into enforceable constructs:

• SQL engines: row filters, column masking, view rewriting

• Storage gateways: path-level allow/deny, token-based checks

• Services/APIs: attribute checks, scoped credentials, purpose gating

Where a surface cannot enforce part of the semantic core, the system must either **fail closed (deny)** or require access through an approved enforcement gateway that can enforce the missing semantics.

3.2.3 Conformance testing (portable semantics)

Portability requires testable guarantees:

- Golden test suite: canonical policy scenarios with expected decisions
- Differential testing: compare decisions across engines for the same inputs
- Capability matrix: declare what each engine supports; block policies that would create semantic divergence

Capability declaration schema: Each enforcement surface MUST register a capability declaration with the control plane:

{

"surface_id": "string",

"surface_type": "sql_engine | gateway | streaming_engine | service | api",

"supported_features": ["row_filter", "column_mask", "deny_override", "purpose_gating", "break_glass_logging"],

"unsupported_features": ["..."],

"capability_version": "string",

"reported_ts": "ISO8601"

}

The unsupported_features field is explicit and mandatory — absence from supported_features alone is not sufficient because it may indicate an outdated declaration rather than a genuine gap.

Failure handling: If a bundle requires a feature not in a surface's supported_features, the rollout controller MUST either: (a) block bundle activation on that surface and route affected tier-1 access through an enforcement gateway, OR (b) reject the bundle with error unsupported_feature and halt the rollout. It MUST NOT silently activate a bundle whose requirements exceed the surface's declared capabilities.

Conformance tests should run in CI for policy changes and periodically in production (e.g., canary probes / sampled scenarios) to detect drift.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.2.4 Rollout and rollback mechanics

- Policies are published as signed bundles; surfaces pull and activate.
- Use progressive rollout (canary → zonal/cluster → regional → global) with automated gates on conformance metrics.
- Ensure rollback is fast and observable (tie into Section 2.5).

Gate override protocol. When an operator needs to override a blocked rollout gate, follow the protocol defined in Section 2.2.4.

3.2.5 Rollback safety invariant

A rollback to a prior bundle version MUST NOT silently re-enable a previously revoked permission. The rollout controller enforces this using the rollback safety invariant defined in Section 2.5.6; the same protocol applies to policy bundle rollbacks in multi-engine deployments.

3.2.6 Operational Acceptance Checks

- Cross-surface conformance pass rate (CI + production probes)
- Policy-version skew window across enforcement points
- Count of "unsupported policy feature" rejections (healthy friction)
- Bypass indicators: access attempts shifting to weaker paths
- Time-to-rollback and rollback success rate

3.3 Evidence-Based Trust Systems

Trust at scale requires audit-grade evidence capture:

- Decision logging: who/what/why under which policy version [7]
- Provenance capture at boundaries (ingest/transform/publish) [5], [9]
- Immutable audit storage with retention controls
- Exception workflows that preserve evidence

3.3.1 Operational acceptance checks:

time-to-evidence (P50/P95); evidence completeness rate; missing-evidence alerts.

3.4 Distributed Governance State for Multi-AZ, DR, and Multi-Region Deployments

Many enterprise lakehouses run primarily in a single region for cost and simplicity, but still require high availability (for example, across availability zones) and frequently operate as multi-instance systems (multiple gateways, multiple clusters, multiple policy evaluators). Some organizations also require cross-region disaster recovery (active-passive), and a smaller subset require active-active or cross-region access.

Governance design should therefore treat "distribution" as a spectrum.

3.4.1 Common deployment models

- Single-region, multi-AZ HA: services survive zonal failures; governance state must remain consistent across multiple in-region enforcement points.
- Single-region, multi-cluster / multi-workspace: multiple engines and gateways in one region; the main risk is semantic divergence and version skew.
- Cross-region DR (active-passive): a secondary region maintains a standby environment with replicated data + governance artifacts for failover.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- Multi-region access / active-active (less common): cross-region access paths exist; governance must handle higher latency and explicit conflict rules.

3.4.2 What must be replicated (beyond data)

Disaster recovery and multi-region access require replicating more than storage objects:

- Dataset registry / catalog metadata (IDs, ownership, classification)
- Policy bundles and contracts (versioned artifacts)
- Evidence configuration and audit-store access controls

In many managed platforms, control-plane components are region-scoped, so DR often involves a separate control-plane instance per region and an explicit sync process.

3.4.3 Consistency and safety semantics (practical defaults)

Artifact types can use different semantics depending on availability requirements:

- Security-critical updates (especially revocations/classification tightening): bounded staleness with prioritized propagation; some deployments use stronger consistency where availability allows.
- Operational metadata: eventual consistency is acceptable only if conflicts are detectable and resolution rules are defined.
- Audit logs: strong integrity guarantees (durable capture with centralized query or verified replication).

Artifact type	Primary risk if stale	Default propagation goal	Conflict handling / authority	Degraded-mode behavior (when control-plane unreachable)	operational acceptance checks
Policy bundle (incl. revocations)	Unauthorized access if revocation is delayed	Bounded staleness; prioritize revocations over grants	Single logical authority; immutable versions; deny overrides allow	Deny-by-default for sensitive scopes; continue last-known-good only within tier TTL	P95 propagation lag (revocations vs grants); version skew across surfaces; rollback success rate
Dataset registry snapshot (IDs, ownership, classification)	Identity drift / wrong dataset classification used	Eventual OK for low tiers; bounded staleness for tier-1	Single authority for canonical dataset ID; conflicts treated as incidents	Freeze registry updates; block publishing new datasets; keep serving last-known-good snapshot with explicit freshness flags	Skew in snapshot IDs; “unknown dataset ID” decision failures; drift alerts
Schema contract artifact	Downstream breakage and	CI-gated publish + staged	Single authority per dataset	Block schema change rollout;	% tier-1 datasets gated; time-to-

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

	silent incompatibility	activation	contract; must reference dataset ID	keep enforcing prior contract; require explicit break-glass for bypass	detect breaking change; bypass rate
Evidence configuration (what must be logged)	Audits non-reproducible; missing evidence	Bounded staleness preferred	Single authority; changes require approval + rationale	If evidence store unreachable, fail closed for regulated scopes or force gateway enforcement	Evidence completeness rate; missing-fields alerts; time-to-evidence (P95)
Decision log schema / integrity controls	Tampering / incomplete capture	Strong integrity guarantees	Durable append-only; centralized query or verified replication	If logging fails, block sensitive actions or require break-glass path	Log drop rate; integrity verification failures; correlation coverage

Table 3. Governance artifact distribution semantics by artifact type.

Table 3 makes consistency, authority, and degraded-mode behavior explicit per artifact type. This prevents implicit assumptions (e.g., 'eventual consistency is fine everywhere') that can create security gaps during partitions and DR events.

3.4.4 Active-passive DR governance model The primary region is the single policy publishing authority. The DR region's control plane operates in read-only mode for artifact publishing during normal operations.

During failover, before routing production traffic:

1. Verify GET /governance_state on DR enforcement surfaces returns a bundle_version within the acceptable skew window.
2. Verify the revocation log is current.
3. If either check fails, a governance operator must provide a written sign-off before traffic cutover; this sign-off is logged as a governance event.

DR drills MUST verify all three conditions. Minimum frequency: quarterly. Results logged and included in governance review.

3.4.5 Operational Acceptance Checks

propagation lag distributions (separately for revocations vs grants); version skew across enforcement points; conflict rate and resolution time; frequency of manual intervention; DR readiness checks (last successful sync time, replay backlog).

3.5 Implementation Decision Matrix and Adoption Roadmap

A phased adoption plan prevents big-bang governance rollouts. Phases should be driven by explicit exit criteria and operational signals (readiness), not calendar time. The sequencing below reflects a common pattern in large organizations but should be adapted to enforcement-surface complexity and risk posture.

Phase -1: Baseline Inventory (prerequisite, not a rollout phase)

Phase 0 exit criteria cannot be measured without completing this first:

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Task	Complete when…
All enforcement surfaces inventoried	A complete list exists: surface type, owner, approximate tier-1 dataset count per surface
Tier assignment process defined	A written ruleset exists for classifying datasets into tier-1 vs best-effort (e.g., sensitivity labels, downstream job count, regulatory scope)
Baseline metrics captured	Current % of decisions with evidence, current schema incident rate, current evidence completeness rate — all documented and owned

Phase -1 is complete when baselines are documented, the tier assignment ruleset is approved, and the surface inventory is current.

Phase 0 — Establish Authority + Contracts + Evidence

Objective: Make governance state *identifiable* and *auditable* for critical datasets before expanding surfaces.

Exit criteria (minimum):

Criterion	Pass Condition	How to Measure
≥90% of tier-1 datasets registered	Registry has all required fields for ≥90% of tier-1 datasets	Query registry: count entries with all required fields / total tier-1 count. Remaining ≤10% must have a remediation date and named owner.
Schema contracts enforced in CI	CI gate blocks merge (not warning) on breaking changes to tier-1 datasets	Attempt a field rename on a tier-1 schema; verify merge is blocked with a structured error
Rollback tested on canary	Automated drill: activate bundle → issue rollback → surface reverts → result logged	Run rollback drill script; verify activation event shows prior bundle_version; verify result in audit store
Evidence capture operational	>99% of tier-1 access decisions include all normative fields	Sample 500 decision log entries; count entries with all required fields present

Failure modes to guard against: shadow catalogs, ad hoc exceptions bypassing logging, schema changes without contract gates.

Phase 1 — Make Governance Distributed + Observable

Objective: Support multi-instance/multi-AZ systems safely, with measurable effective-state correctness.

Exit criteria (minimum):

Criterion	Pass Condition	How to Measure
All tier-1 surfaces implement GET /governance_state	Endpoint returns valid response on all tier-1 production surfaces; non-compliant surfaces listed as 'non-conformant' with remediation date	Query central aggregator surface inventory

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Conformance test suite running	≥50 canonical scenarios running in CI on policy changes; results logged	Check CI pipeline for governance-conformance job
Revocation lag tracked separately	P95 revocation lag tracked separately from grant lag	Aggregator dashboard shows two separate lag histograms
Skew window monitored	Aggregator alerts when any surface exceeds the defined skew window	Trigger a test bundle; verify aggregator raises alert if a surface exceeds the window

Failure modes to guard against: stale caches, partial rollout drift, multi-engine semantic mismatch, unsafe rollback re-enabling access.

Phase 2 — Automation + AI Readiness Controls

Objective: Reduce governance toil and support AI/ML pipelines that reuse data broadly.

Exit criteria (minimum):

Criterion	Pass Condition	How to Measure
DR sync freshness verified	cache_freshness_ts on DR surfaces within defined sync interval (e.g., ≤15 min for policy bundles, ≤5 min for revocations)	Call GET /governance_state on DR surfaces; compare to control plane's last publish timestamp
DR failover drill passed	Drill verifies: latest revocation in DR state; correct bundle_version on DR surfaces; decision logs include correlation IDs	Run quarterly DR drill runbook; log result
AI pipeline surfaces in conformance harness	All surfaces serving AI training, feature stores, or model inference are in the conformance suite and capability matrix	Check conformance harness surface inventory
Audit reconstruction automated	Past decision reconstructed from audit store + decision log without manual steps	Run reconstruction script on a 30-day-old decision; measure time to completion

Failure modes to guard against: AI pipelines bypassing enforcement surfaces, “training reuse” violating classification/purpose, DR failover missing latest revocations.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Decision matrix (how to choose what to implement first)

Prioritize capabilities based on (a) number of enforcement surfaces, (b) regulatory/audit pressure, (c) multi-instance/multi-AZ/DR requirements, and (d) expected rate of schema/policy churn. Platforms with many engines/gateways should prioritize semantic policy core + conformance testing; platforms with DR requirements should prioritize artifact replication + degraded-mode semantics. Table 4 turns the decision matrix into an operator-usable rule-of-thumb for sequencing work without a big-bang rollout.

If your platform has…	Primary failure risk	Implement first	Then add	Operational acceptance check
Many engines/gateways/services	Semantic drift across access paths	Semantic policy core + conformance tests (Section 3.2)	/governance_state + skew/lag SLOs (Section 2.5)	Conformance pass rate stable; drift alerts actionable
High schema churn / many consumers	Schema breakage blast radius	Schema contracts + CI gates (Section 3.1)	Migration windows + rollback protocol	Schema-change incidents and consumer breakages are **measured and attributed** (release tagging), and trend is reviewed over a defined window; any bypasses are **explicitly time-bounded**, approved, and audited.
Regulatory/audit pressure	Evidence gaps and manual audits	Decision logs + immutable audit storage (Section 3.3)	Evidence completeness SLOs	Time-to-evidence (P95) and evidence completeness are **measured**; targets are defined by audit/IR needs; missing-evidence alerts are **actionable** and tracked to closure.
Multi-AZ / multi-instance	Stale caches and partial rollout drift	Effective-state observability + rollout guardrails (Section 2.5)	Bounded staleness + revocation fast path	Version skew and propagation lag are **bounded per tier**; revocation propagation has a defined SLO and is **monitored separately** from

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

				grants.
Cross-region DR / failover	Failover re-enables revoked access	Artifact replication semantics + degraded-mode rules (Section 3.4)	DR drills with governance validation	Failover drill proves "latest revocations applied" and skew visible

Table 4. Decision matrix for sequencing governance capabilities.

Table 4 is a sequencing heuristic, not an empirical benchmark. Teams should adapt thresholds and exit criteria to their risk posture and enforcement-surface complexity.

3.6 Realistic Issues and Implementation Trade-offs

Governance-first design introduces upfront coordination cost but reduces long-term operational burden. Key trade-offs include:

- Coordination now vs stability later
- Evidence capture cost vs audit/incident response speed
- Availability vs consistency under partitions
- Policy portability vs engine-specific expressiveness

IV. Enterprise Implications and AI Readiness

4.1 Governance as AI Platform Foundation

AI amplifies governance requirements because training and feature pipelines reuse data broadly and indirectly. AI readiness depends on dataset identity, provenance, consistent access semantics, and auditable decisions, including the forms of ML-specific technical debt described by Sculley et al. [2] — pipeline glue code, model calibration instability, and retraining dependencies — which compound when the underlying datasets lack stable governance.

4.2 Economic and Operational Benefits

Treating governance as infrastructure shifts work from emergency response to controlled change. Operational metrics to track (and improve over time) include:

- **Lower incident friction:** fewer emergency access exceptions and faster rollback when policy changes misbehave (tracked via exception count and rollback time).
- **Reduced drift:** bounded policy-version skew and shorter revocation propagation lag across enforcement surfaces (tracked via skew/lag SLOs).
- **Faster audits:** improved time-to-evidence retrieval and higher decision-log completeness (tracked via P95 time-to-evidence and missing-field rate).

4.3 Scalability and Future-Proofing

Stable governance semantics allow engine diversity (avoiding "one size fits all" assumptions) [10] and multi-instance expansion without repeated policy rewrites. In many environments, trust scalability becomes a limiting factor alongside peak performance.

V. CONCLUSION

In many enterprise lakehouse implementations, governance becomes the architectural constraint that determines what can scale safely across teams, engines, and regions. Sustainable lakehouses treat governance as a control plane with explicit semantics: authoritative dataset identity, versioned governance artifacts, portable policy evaluation across enforcement surfaces, and audit-grade evidence capture. When the control plane is stable and

observable—policy versions, propagation lag, and drift across engines and regions are measurable—organizations can evolve technology choices without repeatedly reworking security and compliance controls. Contract-based schema evolution and change management reduce coordination overhead between producers and consumers and prevent downstream breakage as reuse scales. A governance-first rollout—establishing identity, contracts, and evidence capture before optimizing performance or expanding engine diversity—avoids governance debt that becomes increasingly expensive to remediate at scale. Finally, AI readiness depends on the same foundations: trustworthy dataset definitions, traceable provenance, and consistent access semantics, so that automated systems can be deployed broadly without sacrificing auditability or control [3], [5], [7], [8].

REFERENCES

[1] Michael Armbrust et al., "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," CIDR 2021. [Online]. Available: https://people.eecs.berkeley.edu/~matei/papers/2021/cidr_lakehouse.pdf

[2] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NeurIPS 2015. [Online]. Available: https://papers.neurips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

[3] M. D. Wilkinson et al., "The FAIR Guiding Principles for scientific data management and stewardship," Scientific Data, 2016. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC4792175/

[4] Vincent C. Hu et al., "Guide to Attribute-Based Access Control (ABAC) Definition and Considerations," NIST Special Publication 800-162, 2014. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf

[5] Luc Moreau et al., "PROV-DM: The PROV Data Model," W3C Recommendation, 2013. [Online]. Available: https://www.w3.org/TR/prov-dm/

[6] Peter Bailis et al., "Probabilistically Bounded Staleness for Practical Partial Quorums," Proc. VLDB Endow. (PVLDB), 2012. [Online]. Available: https://arxiv.org/abs/1204.6082

[7] Karen Kent, Murugiah Souppaya, "Guide to Computer Security Log Management," NIST Special Publication 800-92, 2006. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-92.pdf

[8] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST.AI.100-1, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf

[9] OpenLineage Project, "OpenLineage Documentation (Lineage Collection Specification)," [Online]. Available: https://openlineage.io/docs/

[10] M. Stonebraker and U. Çetintemel, "One Size Fits All: An Idea Whose Time Has Come and Gone," 2005. [Online]. Available: https://cs.brown.edu/~ugur/fits_all.pdf