# Deterministic Governance Kernels for Auditable AI Systems

Mark Randall Havens[*]

*Transparency Ecosphere Project — Independent Research*

Correspondence: beaconwise-tek [at] transparencyecosphere.org | February 2026

**ABSTRACT**

Contemporary AI systems present a fundamental governance problem: their outputs are probabilistic, their internal decision processes are opaque, and the governance mechanisms deployed over them are predominantly shallow policy filters rather than structural enforcement infrastructure. This paper introduces the *deterministic governance kernel* as an architectural pattern for auditable AI systems. A governance kernel is a layer positioned above AI models and automated decision pipelines that enforces deterministic routing rules, maintains cryptographically tamper-evident audit chains, and enables independent deterministic replay of every governed interaction. We describe the architecture, formal invariants, and threat model of BeaconWise — a reference implementation of this pattern currently at v1.9.0 with 355 passing tests and a published open-source specification suite. We argue that the distinction between *governance wrappers* (policy filters applied post-hoc) and *governance kernels* (structural enforcement with verifiable audit continuity) is practically important for regulatory compliance, epistemic integrity, and long-term AI accountability. We describe the kernel's invariants, its threat model against hallucination propagation, hidden persuasion, silent model drift, and governance capture, and situate it relative to existing governance approaches.

*Keywords: AI governance, auditable AI, deterministic systems, cryptographic audit, AI transparency, regulatory compliance, governance infrastructure*

## 1. INTRODUCTION

The governance of AI systems has become a primary concern for regulators, enterprises, and civil society. The EU AI Act, NIST AI RMF, and ISO/IEC 42001 each establish frameworks for managing AI risk, requiring traceability, transparency, and human oversight. Yet a structural gap persists between what these frameworks require and what most deployed governance mechanisms actually provide.

Most AI governance today consists of policy filters — post-hoc rules applied to model outputs to screen for prohibited content, enforce tone constraints, or add disclaimers. These mechanisms share a common failure mode: they govern *what the model says* without governing *how the decision was made*, *whether the audit trail is complete*, or *whether the governance process itself can be independently verified*.

This paper presents an alternative architectural approach: the **deterministic governance kernel**. A governance kernel is not a content filter. It is infrastructure — a layer that enforces deterministic routing, maintains cryptographic evidence continuity, and enables replay-based audit of every governed interaction. The distinction matters because a filter can be bypassed, reconfigured, or silently degraded. A kernel enforces invariants that cannot be overridden by configuration, and its enforcement history is preserved in a tamper-evident chain available for independent verification.

We describe the architecture of BeaconWise, a reference implementation of this pattern. BeaconWise is open-source (Apache 2.0), currently at v1.9.0, and includes a published specification suite comprising nine normative documents covering architecture, security, threat modeling, evidence lifecycle, replay protocol, and validator governance.

### 1.1 Contributions

This paper makes the following contributions. First, we articulate the governance kernel as a formal architectural pattern and define its distinguishing invariants. Second, we present a threat model specific to governance infrastructure — distinct from model robustness or data security — covering hallucination propagation, hidden persuasion, silent drift, and governance capture. Third, we describe the BeaconWise implementation as a concrete, testable instantiation of this pattern. Fourth, we compare the governance kernel approach to existing governance approaches and characterize the design space.

### 1.2 Scope and Limitations

BeaconWise governs the *process* of AI-mediated decision-making — routing, validation, and audit — not the *content* of AI outputs. It does not determine factual truth, moderate speech, or replace domain expertise. The governance guarantees described here apply to the governance layer itself; the probabilistic behavior of underlying AI models remains outside the kernel's determinism boundary. This distinction is elaborated in §3.

## 2. BACKGROUND AND RELATED WORK

### 2.1 The AI Governance Problem

Large language models (LLMs) and related AI systems present governance challenges that differ structurally from earlier software governance problems. Unlike traditional software, LLM outputs are probabilistic: the same input may yield different outputs across calls, making classical software testing insufficient [1]. The opacity of learned representations resists conventional audit: there is typically no machine-readable account of why a particular output was generated [2,3].

These properties create three distinct audit failures. First, *post-hoc irrecoverability*: once an AI interaction occurs without governance recording, the decision basis is unrecoverable. Second, *silent drift*: model behavior can change across API versions, fine-tuning updates, or retrieval corpus changes without any observable governance event. Third, *governance theater*: policy filters can be satisfied in ways that technically comply with rules while violating their intent [4].

### 2.2 Existing Governance Approaches

Contemporary AI governance mechanisms fall into several categories. *Constitutional AI* [5] and related RLHF-based approaches train governance preferences into model weights. These improve alignment but do not provide external audit continuity; the model's governance behavior is not independently verifiable from outside the model.

*Guardrails systems* such as Nvidia NeMo Guardrails [6], Llama Guard [7], and similar tools apply rule-based or classifier-based filters to model I/O. These provide policy enforcement but typically produce no audit records, operate non-deterministically, and cannot be independently replayed.

*Evaluation frameworks* such as HELM [8], MMLU [9], and safety benchmarks assess model capabilities and behaviors in test settings. These provide deployment decision support but do not govern operational interactions.

*Logging and observability* systems record model interactions for monitoring but typically without cryptographic integrity guarantees, deterministic replay capability, or governance decision traceability.

No existing system provides all of: (a) deterministic routing with verifiable invariants, (b) cryptographically tamper-evident audit chains, (c) deterministic replay with explicit divergence classification, and (d) governance independence from model providers. The governance kernel is designed to provide these properties together.

### 2.3 Regulatory Context

The EU AI Act (Regulation 2024/1689) imposes technical documentation, traceability, logging, and human oversight requirements on high-risk AI systems [10]. The NIST AI RMF organizes governance activities across Govern, Map, Measure, and Manage functions [11]. ISO/IEC 42001:2023 defines AI management system requirements including audit, traceability, and continual improvement [12]. These frameworks converge on a set of infrastructure capabilities — reproducible audit records, verifiable governance decisions, and accountable oversight chains — that policy filters do not provide. The governance kernel directly targets this infrastructure gap.

## 3. GOVERNANCE KERNEL ARCHITECTURE

A deterministic governance kernel is a layer positioned above AI inference systems and below their consumers. It does not modify AI outputs. It enforces governance rules over the process by which outputs are validated, routed, and recorded, and it maintains a tamper-evident audit history of every governed interaction. The BeaconWise Transparency Ecosphere Kernel (TEK) is a concrete implementation of this pattern.

### 3.1 Architectural Layers

BeaconWise organizes governance across six layers:

| Layer | Function |
|---|---|
| Inference Interface | Captures I/O; normalizes metadata; records routing decisions. Does not modify outputs. |
| Validation Layer | Independent evaluation of integrity, determinism, and policy compliance. |
| Evidence Lifecycle | EPACK chain formation; SHA-256 hash binding; append-only persistence. |
| Governance Layer | Constitutional invariant enforcement; anti-capture controls. |
| Challenger Layer | Independent escalation; replay verification authority. |
| Audit / Replay | Deterministic replay; divergence detection; certificate generation. |

**Figure 1: BeaconWise Governance Kernel Architecture**

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ CONSUMER APPLICATION LAYER ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ GOVERNANCE KERNEL (BeaconWise TEK v1.9.0) ■
■ ■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■
■ ■ Inference I/F ■→ ■ Evidence Validation ■ ■
■ ■ (capture I/O) ■ ■ Gate (I2 enforced) ■ ■
■ ■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■
■ ■■■■■■■■■■■■■■■■■■■■ ↓ ■
■ ■ Routing Engine ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■
■ ■ (deterministic) ■ ■ EPACK Chain (SHA-256) ■ ■
■ ■ BLOCK■DEFER ■ ■ append-only · tamper ■ ■
■ ■ CONFIRM■PLAN ■ ■ evident · replayable ■ ■
■ ■ PROCEED ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■
■ ■■■■■■■■■■■■■■■■■■■■ ■
■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■
■ ■ Validator Consensus + Challenger Layer ■ ■
■ ■ (multi-validator · anti-capture · auditable) ■ ■
■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ AI MODEL LAYER (OpenAI / Anthropic / OSS) ■
■ ZERO-TRUST STANCE ← Invariant I4 ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

The governance kernel intercepts all AI model I/O. Routing is deterministic (Invariant I1). Evidence validation gates reject unvalidated output (I2). The EPACK chain cryptographically binds every interaction record (I5). Validator consensus and the Challenger layer enforce governance independence (I6, I7). All layer boundaries are crossed only through kernel-controlled interfaces; no consumer application has direct model access.

### 3.2 Core Invariants

The kernel enforces eight constitutional invariants. These are normative — they cannot be overridden by configuration or operator instruction:

| ID | Invariant | Specification |
|---|---|---|

| | | |
|---|---|---|
| I1 | **Deterministic Routing** | Given identical inputs and system state, routing decisions (BLOCK / DEFER / CONFIRM / PLAN / PROCEED) are identical across all invocations. |
| I2 | **Evidence Validation Gates** | All AI outputs must pass structural validation before delivery. Failed validation produces a structured safe fallback — never unvalidated output. |
| I3 | **Audit Replayability** | Every interaction produces a complete replay package enabling independent reconstruction of the governance decision. |
| I4 | **Zero-Trust Model Stance** | All AI outputs are treated as untrusted until validation pipeline completion. No model provider has bypass access. |
| I5 | **Cryptographic Traceability** | Each EPACK carries a SHA-256 hash referencing the preceding EPACK, forming an append-only tamper-evident chain. |
| I6 | **Validator Independence** | Governance validation is performed by entities independent from AI model providers and deployment operators. |
| I7 | **Failure Transparency** | Indeterminate or unsafe governance states produce explicit disclosure artifacts. The kernel never silently defaults to permissive behavior. |
| I8 | **Constitutional Non-Override** | Invariants I1–I8 cannot be suspended by configuration. Amendments require documented public justification. |

### *3.3 Deterministic Routing*

The routing layer classifies each governed interaction into one of five outcome classes: BLOCK (harmful input detected), DEFER (input exceeds competence threshold), CONFIRM (human confirmation required before delivery), PLAN (multi-step execution required), or PROCEED (generate and validate). Classification uses deterministic rules applied over: (a) safety screening results from the input classifier, (b) belief state across five knowledge domains maintained by Bayesian inference, (c) policy configuration, and (d) domain-specific routing tables. The routing decision is invariant: the same input and system state always yield the same classification. This property is machine-testable and constitutes the core auditability guarantee.

### *3.4 Evidence Packaging (EPACK)*

Every governed interaction produces an Evidence PACKet (EPACK) — an atomic audit record with the following required fields: input payload hash, governance configuration snapshot, routing decision with rationale, validation results, validator attribution, environment fingerprint, and SHA-256 hash of the preceding EPACK. EPACKs are append-only; no modification is permitted after sealing. Any post-hoc modification breaks the hash chain and is detectable by any party with access to the replay engine, without requiring access to the original system. This property provides cryptographic tamper evidence without requiring a blockchain or external trust anchor.

### *3.5 Deterministic Replay Protocol*

The Deterministic Replay Protocol (DRP) specifies how any governed run can be reconstructed from its Replay Package (RP). A replay proceeds in three steps. First, integrity verification: EPACK chain links, input hashes, and validator output hashes are verified against recorded values. Any failure terminates replay with REPLAY_RESULT = TAMPER_DETECTED. Second, environment equivalence check: the current replay environment fingerprint is compared against the recorded one; differences are classified as drift factors and the replay continues with DRIFT_RISK status rather than terminating. Third, deterministic execution: governance decisions are reproduced using the recorded routing decisions and determinism policy. The replay result is either VERIFIED (identical), DRIFT (explainable divergence), or TAMPER_DETECTED (integrity failure). Silent divergence is not a permitted outcome — the protocol enforces explicit classification.

### *3.6 Validator Governance*

BeaconWise employs a multi-validator consensus architecture with a Challenger layer providing independent oversight. The Validator Governance Constitution prohibits single-entity validator control, requires auditable validator identities, and mandates independent operation of secondary validators from primary validators. The Challenger layer — a role distinct from validators — has authority to trigger replay audits, escalate disputes, and initiate additional validation independent of the primary consensus process. These structural anti-capture provisions distinguish the governance kernel from logging systems, which typically have no mechanism for detecting validator compromise.

### *3.7 Kernel Boundary Specification*

Precisely defining what lies outside the governance kernel's scope is as important as defining what lies within it. The following properties are explicitly **outside** the kernel's determinism boundary: AI model output generation (probabilistic), factual accuracy determination (domain expert responsibility), content moderation policy (organizational policy), legal compliance determination (legal analysis), and user interface design (application layer).

The following properties are explicitly **inside** the kernel's determinism boundary: routing classification given a defined system state, evidence validation gate pass/fail determination, EPACK record generation and hash chain formation, validator consensus computation, replay execution and divergence classification, and constitutional invariant enforcement. This boundary is the governance kernel's fundamental contribution: it makes the governance process deterministic and auditable even when operating over a probabilistic AI substrate.

## 4. GOVERNANCE THREAT MODEL

Governance infrastructure introduces a threat surface distinct from model robustness or infrastructure security. The relevant threats are not primarily adversarial model inputs (jailbreaks, prompt injection) but attacks on the governance layer itself — mechanisms that would allow a system to appear governed while actually being ungoverned. We identify four primary threat categories relevant to the design of governance kernels.

### *4.1 Hallucination Propagation*

AI language models generate plausible-sounding outputs that may be factually incorrect, unsupported by cited sources, or internally inconsistent — a property commonly termed hallucination [13,14]. From a governance perspective, hallucination is not primarily a model accuracy problem but an audit traceability problem: when a hallucinated output is acted upon, there is typically no record of which evidence supported the output, what validation was applied, or whether the output was flagged as uncertain.

BeaconWise addresses this via the evidence validation gate (Invariant I2) combined with EPACK-recorded evidence provenance. The validation gate requires that outputs pass structural integrity and schema compliance checks before delivery; outputs that fail produce safe fallback artifacts with explicit failure codes rather than unvalidated content. EPACK records capture the evidence sources used in generation with cryptographic provenance, enabling post-hoc audit of what information supported each governed output. This does not prevent hallucination but creates an auditable record of it.

### *4.2 Hidden Persuasion*

AI systems optimized for engagement or user satisfaction may exhibit behaviors that systematically bias user beliefs or decisions — sycophancy, framing effects, selective information presentation — without any observable governance event [15]. This is structurally different from content policy violations because the behavior may comply with all explicit rules while systematically undermining user epistemic autonomy.

BeaconWise addresses this through a constitutional prohibition on persuasion optimization (Transparency Over Persuasion principle) and the Transparent System Voice (TSV) framework, which reduces anthropomorphic framing that enables exploitation of human social cognition. The EPACK record preserves the governance state at delivery time, enabling retrospective audit of whether persuasive patterns emerged over interaction sequences.

### *4.3 Silent Model Drift*

AI model behavior can change without observable governance events through model updates, fine-tuning, retrieval corpus changes, or context window drift [16]. From a compliance perspective, this creates a documentation gap: the system behaves differently than the documented version, but no governance record captures the change.

Deterministic replay provides structural detection of model drift. Because routing decisions are deterministic and recorded, replaying a historical interaction against current system state detects any behavioral change as an explicit divergence — classified as DRIFT with an associated drift factor record — rather than silent behavioral change. This does not prevent drift but ensures it is observable and documented.

### *4.4 Governance Capture*

Governance systems are themselves subject to capture — a condition where oversight authority becomes concentrated, opaque, or aligned with the interests of the governed entity rather than independent oversight. Capture risk is well-documented in regulatory contexts [17] but rarely modeled explicitly in AI governance architecture.

BeaconWise models governance capture as an explicit threat category and implements structural mitigations: validator independence requirements, anti-capture provisions in the constitutional governance charter, the Challenger layer as an oversight body independent of the validator consensus, and prohibition on single-entity validator control. The governance constitution itself is public and versioned, making any governance degradation observable to external parties.

| Threat Class | Attack Vector | Kernel Mechanism | Invariant |
|---|---|---|---|
| Hallucination Propagation | Unvalidated AI output reaches consumers without evidence provenance | Evidence Validation Gate + EPACK evidence provenance recording | I2, I5 |
| Hidden Persuasion | AI optimized for engagement exploits human social cognition | Constitutional Transparency Over Persuasion prohibition + TSV framework | I7 (constitutional) |
| Silent Model Drift | Model behavior changes across versions without governance event | Deterministic replay detects divergence; explicit DRIFT classification — never silent | I1, I3 |
| Non-Reproducible Outputs | Governance decisions cannot be independently reconstructed | Deterministic Replay Protocol produces VERIFIED / DRIFT / TAMPER_DETECTED — no silent outcome | I1, I3, I5 |
| Governance Capture | Validator authority concentrates or becomes provider-aligned | Multi-validator consensus + Challenger layer + constitutional anti-capture provisions + public versioned charter | I6, I8 |

*Table 3: Governance threat model — threat classes, attack vectors, and kernel mitigations.*

## 5. COMPARISON TO EXISTING GOVERNANCE APPROACHES

The governance kernel pattern differs from existing approaches along three key dimensions: architectural depth (wrapper vs. infrastructure), verification mode (evaluation vs. enforcement), and audit continuity (logging vs. tamper-evident chain). Table 1 summarizes these distinctions.

| Approach | Architectural Depth | Audit Continuity | Replay Capability | Governance Independence |
|---|---|---|---|---|
| Constitutional AI / RLHF | Model-internal (weights) | None — behavior not externally auditable | Not applicable | Not applicable — model provider controls |
| Guardrails / Policy Filters (NeMo, Llama Guard) | Post-hoc wrapper | No cryptographic guarantee; typically no audit record | Not supported | Often tightly coupled to inference provider |

| | | | | |
|---|---|---|---|---|
| Evaluation Frameworks (HELM, benchmarks) | Pre-deployment assessment | Deployment-time snapshot; no operational continuity | Not applicable | Independent of model provider |
| Observability / Logging (LLM observability tools) | Infrastructure layer | Records interactions; no cryptographic integrity | Limited; no determinism guarantee | Variable; often provider -coupled |
| **Governance Kernel (BeaconWise)** | **Independent governance infrastructure** | **SHA-256 EPACK chain; tamper-evident; append-only** | **Full deterministic replay; explicit divergence classification** | **Constitution ally enforced provider ind ependence** |

Table 1: Comparison of AI governance approaches across key dimensions. BeaconWise row represents the governance kernel pattern.

The critical distinction between wrappers and infrastructure is not merely technical sophistication but the nature of the governance guarantee. A wrapper intercepts outputs and applies rules; its operation can be monitored, but its governance behavior is not independently verifiable by external parties without access to the deployment environment. Infrastructure enforces invariants whose compliance can be verified from the audit record alone, without access to the running system. This property — sometimes called *audit independence* — is what regulatory frameworks require when they specify that AI systems be auditable by third parties.

The distinction between evaluation and enforcement is similarly consequential. Evaluation approaches produce evidence about model behavior under test conditions but do not govern operational behavior. A model may score well on safety benchmarks while exhibiting unsafe behavior in operational contexts, a gap documented in multiple evaluation studies [18,19]. Enforcement approaches apply governance rules to every operational interaction, not to test samples.

## 6. IMPLEMENTATION: BEACONWISE V1.9.0

BeaconWise is a reference implementation of the governance kernel pattern, developed as open-source infrastructure under the Apache 2.0 license. As of v1.9.0 (February 2026), the implementation includes the following components:

| Component | Status | Key Metric |
|---|---|---|
| Governance kernel (routing, validation) | Production | Deterministic routing across 5 outcome classes; 100% routing test coverage |
| EPACK chain (evidence lifecycle) | Production | SHA-256 chain with 6-stage lifecycle: ingest → validate → bind → persist → challenge → archive |
| Replay engine (DRP implementation) | Production | Full RP reconstruction; VERIFIED / DRIFT / TAMPER_DETECTED classification |
| Validator consensus + Challenger layer | Production | Multi-validator; constitutionally enforced independence; challenger escalation |
| Safety screening (input classification) | Production | Deterministic rule-based classifier; no probabilistic classification in governance path |
| LLM provider adapters | Production | OpenAI, Anthropic, open-source models, retrieval pipelines; vendor-neutral |
| Test suite | 355 passing tests | Governance kernel, replay, validator consensus, evidence lifecycle, V9 capabilities |
| Specification suite | 9 normative docs | Architecture, Security, Threat Model, Replay Protocol, Evidence Lifecycle, Validator Governance, Compliance Mapping, Constitution, Adoption Guide |

Table 2: BeaconWise v1.9.0 component status.

The architecture is organized around the src/ecosphere/ package hierarchy, with submodules for the kernel, consensus, governance, replay, validation, evidence lifecycle (epack), safety screening, provider adapters, and the Transparent System Voice (TSV) framework. The governance constitution is machine-readable and normatively referenced by the kernel at runtime — constitutional invariants are not documentation but enforced constraints.

### *6.1 Verification Properties*

Each constitutional invariant is associated with a specific verification procedure. Invariant I1 (Deterministic Routing) can be verified by any party by replaying the same input N times against the stored EPACK records and confirming identical routing decisions in all N cases. Invariant I5 (Cryptographic Traceability) can be verified by any party with access to the EPACK chain by confirming that each record's hash correctly references its predecessor — a procedure that requires no access to the original system or private keys. Invariant I7 (Failure Transparency) can be verified by injecting ambiguous or safety-indeterminate inputs and confirming that explicit disclosure artifacts are produced rather than silent passthrough. These properties make BeaconWise governance externally auditable in the sense required by regulatory frameworks: a third party can verify governance compliance from the audit record alone.

### *6.2 Open Source Availability*

The complete source code, specification suite, and test suite are publicly available under Apache 2.0. The governance constitution, threat model, and all normative specifications are published as versioned documentation within the repository. The specification documents use RFC 2119 normative language (MUST, SHOULD, MAY) and are designed to be machine-readable for future automated conformance verification.

**Figure 2: BeaconWise v1.9.0 Package Structure**

```
src/ecosphere/
■■■ kernel/ # Constitutional invariant enforcement
■ ■■■ router.py # Deterministic routing (I1)
■ ■■■ gates.py # Evidence validation gates (I2)
■ ■■■ constitution.py # 13 invariants, runtime-enforced
■■■ consensus/ # Multi-validator consensus (I6)
■ ■■■ validators.py # Primary + secondary validators
■ ■■■ challenger.py # Independent oversight layer
■■■ governance/ # Policy routing tables
■■■ replay/ # DRP implementation (I3)
■ ■■■ engine.py # RP reconstruction + classification
■ ■■■ certificates/ # VERIFIED/DRIFT/TAMPER_DETECTED
■■■ epack/ # Evidence lifecycle (I5)
■ ■■■ chain.py # SHA-256 append-only hash chain
■ ■■■ lifecycle.py # ingest→validate→bind→persist→archive
■■■ safety/ # Deterministic input classification
■■■ providers/ # Vendor-neutral adapters
■ ■■■ openai.py # OpenAI adapter
■ ■■■ anthropic.py # Anthropic adapter
■ ■■■ retrieval.py # RAG/retrieval pipelines
■■■ tsv/ # Transparent System Voice framework
docs/ # 9 normative specification documents
tests/ # 355 passing tests (full invariant coverage)
```

*Package structure as of v1.9.0. All modules enforce constitutional invariants through kernel interfaces; no module has direct model provider access.*

### *6.3 Regulatory Infrastructure Mapping*

Table 4 maps BeaconWise implementation components to specific regulatory infrastructure requirements. This mapping is informative only; compliance requires organizational processes and legal analysis beyond governance infrastructure capabilities.

| Regulatory Requirement | Standard / Article | BeaconWise Component | Strength |
|---|---|---|---|
| Automatic log generation for each operation | EU AI Act Art. 12 | EPACK mandatory audit recording | Direct implementation |

| | | | |
|---|---|---|---|
| Technical documentation of system capabilities | EU AI Act Art. 11 | 9 normative public specifications + ARCHITECTURE.md | Substantive |
| Human oversight capability | EU AI Act Art. 14 | CONFIRM routing gate; Challenger escalation | Infrastructure support |
| Measurement of AI system behavior | NIST AI RMF MEASURE 2.5 | Deterministic replay enables regression testing across versions | Direct implementation |
| Incident response and documentation | NIST AI RMF MANAGE 1.3 | EPACK incident recording with permanent audit entry | Substantive |
| Internal audit capability | ISO/IEC 42001 Cl. 9.2 | EPACK chain replay without access to production system | Substantive |
| Risk management documentation | ISO/IEC 42001 Cl. 6.1 | Formal threat model with 5 adversary classes | Strong |
| AI policy documentation | ISO/IEC 42001 Cl. 5.2 | Machine-readable CONSTITUTION.md with 13 normative invariants | Strong |

*Table 4: BeaconWise regulatory infrastructure mapping. Deployers remain responsible for full compliance; this table maps technical capabilities to regulatory infrastructure prerequisites only.*

### *6.4 V9 Resilience Control Plane*

BeaconWise v1.9.0 extends the governance kernel with a Resilience Control Plane — a closed-loop feedback architecture that closes the gap between governance anomaly detection and recovery action. Where earlier versions could detect governance failures (via EPACK replay and tamper classification), v1.9.0 can also respond deterministically, verify that responses improved system health, and circuit-break persistently degraded validators — all with full EPACK audit continuity.

The control plane is implemented across nine components in the meta_validation/ module. The **TSI Tracker** maintains a sliding-window Trust-Signal Index aggregating interaction outcomes (PASS=0.90, WARN=0.70, REFUSE=0.45, ERROR=0.30) with exponential decay weighting and a 15-minute linear forecast — replacing the prior hardcoded 0.85/0.55 TSI thresholds with a runtime-computed signal. The **Recovery Engine** performs deterministic plan selection over a tiered recovery policy compiled from YAML, applying budget constraints (latency, cost), tier penalties, and oscillation penalties, with tie-breaking on (score, predicted_independence_gain, −tier). The **Damping Stabilizer** applies PID-inspired rollout velocity control ($k_p$=0.5, $k_i$=0.2, $k_d$=0.1) to prevent governance oscillation — the yo-yo pattern where rapid recovery actions introduce more instability than the original failure.

The **Circuit Breaker** tracks per-plan failure sequences through a CLOSED → OPEN → HALF_OPEN → CLOSED state machine, with auditable state_snapshot() output persisted to EPACK. The **Post-Recovery Verifier** closes the loop: after a recovery action is applied, it checks whether TSI actually improved against configured thresholds and produces a structured rollback recommendation if it did not. The **Meta-Validation Index (MVI)** operationalizes the governance kernel principle that governance must itself be governed: it computes a weighted composite score (replay stability 40%, recovery consistency 35%, TSI coherence 25%, pass threshold 0.80) as a runtime health indicator for the governance pipeline itself — not for AI output quality.

All resilience events — RECOVERY_TRIGGERED, RECOVERY_DECISION, RECOVERY_APPLIED, RECOVERY_VERIFIED, RECOVERY_ROLLBACK, CIRCUIT_BREAKER — are recorded as **Recovery EPACK Events**, hash-chained via prev_hash and persisted to the standard EPACK audit store. Resilience governance is therefore auditable by the same replay infrastructure as any other governed interaction. The **Policy Compiler** compiles the resilience_policy YAML block (enterprise_v9.yaml) into a ResilienceRuntime instance at startup, with graceful degradation on parse errors. The **Resilience Runtime** provides the orchestration API: maybe_recover() → engine decide → damping → circuit breaker; verify_recovery() → post-recovery TSI check → circuit breaker feedback; record_outcome() → TSI tracker update.

| Component | Module | Core Property |
|---|---|---|
| TSI Tracker | meta_validation/tsi_tracker.py | Sliding-window trust-signal with exponential decay; 15-min forecast; replaces hardcoded thresholds |
| Recovery Engine | meta_validation/recovery_engine.py | Deterministic plan selection over tiered policy; budget + oscillation constraints; audit-logged decisions |
| Damping Stabilizer | meta_validation/damping_stabilizer.py | PID-inspired rollout control (kp=0.5, ki=0.2, kd=0.1); prevents recovery oscillation; integral capping |
| Circuit Breaker | meta_validation/circuit_breaker.py | Per-plan CLOSED→OPEN→HALF_OPEN state machine; state_snapshot() → EPACK; manual break-glass reset |
| Post-Recovery Verifier | meta_validation/post_recovery_verifier.py | Closed-loop: did recovery improve TSI? MVI check + structured rollback recommendation |
| Meta-Validation Index | meta_validation/mvi.py | Validate the validator: replay stability (40%) + recovery consistency (35%) + TSI coherence (25%) |
| Recovery EPACK Events | meta_validation/recovery_events.py | 6 event types; prev_hash chained; persisted to EPACK JSONL + in-memory store |
| Policy Compiler | meta_validation/policy_compiler.py | Compiles resilience_policy YAML → ResilienceRuntime; V8/V9 policy shapes; graceful degradation |
| Resilience Runtime | meta_validation/resilience_runtime.py | Orchestration: maybe_recover() + verify_recovery() + record_outcome() + dependency_metrics() |

*Table 5: V9 Resilience Control Plane components. All components are deterministic, independently testable, and produce EPACK-persisted audit records.*

The Resilience Control Plane is significant for governance credibility because it operationalizes the distinction between detection and response. A governance kernel that detects anomalies but cannot respond to them provides audit continuity without remediation capability — useful for forensic purposes but insufficient for operational governance in production environments. The V9 control plane closes this gap while preserving the core invariant: every recovery decision is deterministic, every action is auditable, and every outcome is verifiable. Resilience governance is not a parallel system; it is an extension of the same EPACK audit infrastructure that governs AI inference.

## 7. DISCUSSION

### *7.1 Governance Kernels as Infrastructure*

The analogy between governance kernels and operating system kernels is instructive. An OS kernel does not constrain application logic; it enforces resource management, memory protection, and process isolation as non-negotiable structural properties of the execution environment. Applications run on top of the kernel but cannot violate its invariants through configuration. The governance kernel applies the same pattern to AI governance: AI models operate within the governance environment but cannot bypass its enforcement through output manipulation or deployment configuration.

This framing suggests that AI governance infrastructure will likely follow a layered adoption path similar to other infrastructure standards: initial adoption by early-mover regulated industries (healthcare, finance, legal), followed by expansion as regulatory requirements formalize, followed by commodity adoption as governance infrastructure becomes assumed baseline. The current moment — regulatory frameworks active but implementation requirements underspecified — is the appropriate window for establishing open governance infrastructure standards.

### *7.2 Determinism Scope*

A critical clarification: the determinism guarantee in a governance kernel applies to the *governance decision*, not to the AI model output. Underlying LLMs remain stochastic; the same prompt may yield different responses. The governance kernel's determinism boundary covers the routing, validation, and audit recording processes — which are deterministic — and explicitly records which aspects of the system are non-deterministic. This is the appropriate scope for regulatory purposes:

regulators need to verify that governance processes were consistently applied, not that AI outputs were identical.

### *7.3 Limitations*

BeaconWise addresses the governance infrastructure problem but does not address the full scope of AI governance. Evidence validation gates check structural integrity and policy compliance, not factual accuracy. Cryptographic audit chains preserve governance history but cannot recover from catastrophic storage loss without redundancy. Validator independence provisions prevent formal capture but cannot prevent informal influence. Anti-persuasion provisions address architectural design choices but cannot prevent persuasive content from appearing in AI outputs.

More fundamentally, a governance kernel governs the *process* of AI-mediated interaction, not the *quality* of AI judgment. An AI system producing consistently wrong answers will have a well-governed audit trail of consistently wrong answers. The governance kernel is necessary but not sufficient for trustworthy AI systems.

### *7.4 Relationship to Regulatory Requirements*

The governance kernel architecture maps directly to several infrastructure requirements that existing governance frameworks specify but do not provide. The EU AI Act Article 12 requires that high-risk AI systems 'automatically generate logs'; EPACK mandatory audit recording satisfies this at the infrastructure level rather than relying on application-layer logging. NIST AI RMF MEASURE 2.5 requires that AI systems be testable; deterministic replay enables regression testing of governance decisions across software versions. ISO/IEC 42001 Clause 9.2 requires internal audit capability; EPACK chain replay enables third-party audit without requiring access to the production system.

This mapping does not imply that deploying BeaconWise satisfies regulatory obligations. Compliance requires organizational processes, legal analysis, and domain-specific governance that governance infrastructure alone cannot provide. The claim is narrower: governance infrastructure that lacks the properties provided by the governance kernel — tamper-evident audit chains, deterministic replay, invariant enforcement — cannot satisfy the infrastructure prerequisites that these regulatory requirements assume.

### *7.5 Future Work*

Several directions merit further development. First, formal verification of governance invariants: the current invariants are specified in natural language and tested empirically; formal specifications amenable to automated proof would strengthen the governance guarantee. Second, evaluation methodology for governance kernels: a benchmark for comparing governance infrastructure systems across auditability, determinism, and replay fidelity dimensions would support the emerging field. Third, sector-specific governance profiles: the generic kernel architecture needs instantiation in healthcare, legal, and financial contexts where domain-specific governance requirements interact with the general infrastructure properties described here. Fourth, interoperability: as governance infrastructure matures, standardization of audit record formats and replay protocols would enable cross-system governance auditing.

## 8. CONCLUSION

We have presented the deterministic governance kernel as an architectural pattern for auditable AI systems, characterized its distinguishing invariants, described its threat model, and documented the BeaconWise TEK as a reference implementation. The central claim is straightforward: effective AI governance requires infrastructure — deterministic enforcement, cryptographic audit continuity, and independent replay capability — that existing governance approaches do not provide.

The BeaconWise implementation demonstrates that these properties are achievable in a production-grade open-source system. The 355-test suite provides a verifiable conformance baseline. The nine normative specification documents provide a foundation for standardization. The Apache 2.0 license ensures the infrastructure remains available as a commons resource rather than a proprietary governance layer.

AI governance is at an inflection point. Regulatory frameworks are active; implementation requirements are being defined; the infrastructure layer that makes compliance operationally tractable is nascent. The governance kernel pattern, and BeaconWise as a reference implementation, represents one contribution to that infrastructure foundation.

## REFERENCES

[1] Bommasani, R. et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[2] Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

[3] Lipton, Z. C. (2018). The mythos of model interpretability. *Queue, 16(3)*, 31–57.

[4] Perez, E. et al. (2022). Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

[5] Bai, Y. et al. (2022). Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*.

[6] Rebedea, T. et al. (2023). NeMo Guardrails: A toolkit for controllable and safe LLM applications. *Proceedings of EMNLP 2023*.

[7] Inan, H. et al. (2023). Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*.

[8] Liang, P. et al. (2022). Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

[9] Hendrycks, D. et al. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

[10] European Parliament (2024). Regulation 2024/1689 on artificial intelligence. *Official Journal of the European Union*.

[11] NIST (2023). AI Risk Management Framework (AI RMF 1.0). *National Institute of Standards and Technology*. doi:10.6028/NIST.AI.100-1.

[12] ISO/IEC (2023). ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system. *International Organization for Standardization*.

[13] Maynez, J. et al. (2020). On faithfulness and factuality in abstractive summarization. *Proceedings of ACL 2020*.

[14] Ji, Z. et al. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55(12)*, 1–38.

[15] Perez, E. et al. (2023). Sycophancy to subterfuge: Investigating reward tampering in language models. *arXiv preprint arXiv:2306.09467*.

[16] Chen, L. et al. (2023). How is ChatGPT's behavior changing over time? *arXiv preprint arXiv:2307.09009*.

[17] Stigler, G. J. (1971). The theory of economic regulation. *The Bell Journal of Economics and Management Science, 2(1)*, 3–21.

[18] Huang, J. et al. (2023). A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

[19] Kambhampati, S. (2024). Can LLMs really reason and plan? *Communications of the ACM*.

[*] Correspondence: Transparency Ecosphere Project, Louisville, KY. Source code and specifications: https://github.com/beaconwise-tek/beaconwise (verify current URL) License: Apache 2.0 Submitted: February 2026