The Distribution Drift Bound

A Formal Derivation of MAI-1 Invariant 3: Threshold Calibration, Online Detection, and Inference-Time Tractability

Clause AI-6

Auburn Patent Family

Fields

Specification — Version 1.0

February 2026

Abstract

The deployment of foundation models in regulated industries—financial services, healthcare, defense, and critical infrastructure—demands continuous verification that a model's operational behavior remains consistent with the distribution under which it was validated. Static benchmark evaluations performed prior to deployment provide no guarantee of continued adherence to baseline performance in non-stationary production environments. The Model Attestation Interface (MAI-1; Clause AI-5, Auburn Patent Family; Fields, 2026) defines five mandatory health invariants, of which Invariant 3—Distribution Drift—requires that the Kullback–Leibler divergence between the model's current output distribution and its validated baseline remain below a certified ceiling at all attested measurement points. MAI-1 defines the `drift-kl` field, its governance rationale, and its breach semantics, but does not derive the bound, specify the calibration methodology, or define the baseline distribution against which drift is measured.

This document fills that gap. It provides the complete mathematical foundation for MAI-1's distribution drift invariant: the information-theoretic properties of KL divergence in the high-dimensional output and hidden-state spaces of modern Large Language Models; the formal taxonomy of dataset shift (covariate, prior probability, and concept drift) adapted to generative AI pathologies including coherence collapse and planning drift; the baseline distribution construction methodology using probabilistic sketch representations (Count-Min Sketch, HyperLogLog) distributed via IETF Concise Reference Integrity Manifests (CoRIM); the online drift detection algorithm suite (ADWIN, Page-Hinkley, KSWIN) with formal justification for algorithm selection and parameter calibration; the conformal prediction integration that provides distribution-free uncertainty quantification on drift estimates and connects directly to commercial insurability (Munich Re aiSure); and the computational optimization strategy—logit projection, Top-K truncation, asynchronous sampling, and variational surrogate models—that makes inference-time KL divergence monitoring feasible at scale within a 1–2% latency budget.

The result is a governance-grade specification that transforms the qualitative concern of "behavioral drift" into an auditable, insurable, and enforceable property of the deployed model. With Clause AI-6 enforced alongside AI-8 (Entropy Collapse Constraint) and AI-2 (Gradient Starvation Envelope), three of the five mandatory MAI-1 invariants are formally derived with full calibration methodology, completing the analytical core of the Auburn Governance Stack's Layer 2 invariant suite.

Contents

Implementation Risk Notice

This document is published in full under CC BY-NC-ND 4.0 for academic transparency and regulatory reference. Publication of the complete specification does not constitute implementation guidance. Incorrect implementation of the methods described herein poses material risk to model stability, compliance posture, and commercial insurability.

Dependency Architecture

Clause AI-6 does not operate in isolation. It is the fourth mandatory invariant in the MAI-1 attestation architecture and depends on, feeds into, and is constrained by multiple documents in the Auburn Governance Stack that are not contained in this specification:

- **MSAF (Model State Attestation Framework):** Defines the three-tier attestation architecture and the model health invariant structure that Clause AI-6 implements. Without MSAF's architectural constraints, the drift invariant lacks its cryptographic binding to hardware attestation (Layer 1) and provenance (Layer 3). An implementation that monitors drift without the MSAF binding produces *unsigned evidence*—metrics without attestation, which are inadmissible for governance purposes.
- **MAI-1 (Model Attestation Interface):** Normatively defines Invariant 3 (Distribution Drift) at §7.3, specifying the `drift-kl` and `thresholds.drift-kl-max` fields in the Layer 2 payload. Clause AI-6 derives the bound and calibration methodology, but the *payload format*, *signing protocol*, *versioning policy*, and *inter-invariant composition rules* are defined in MAI-1. An implementation of Clause AI-6 that does not conform to MAI-1's payload specification produces evidence that no MAI-1-conformant Verifier can appraise.
- **CTS-1 (Conformance Test Suite):** Defines the binary pass/fail assertions (AS-SM-L2-10 through TE-SM-L2-10.04) that test Clause AI-6 conformance. The threshold calibration methodology in Section 6.3, the state machine transition rules in Section 6.4, and the detection pipeline decision logic in Section 7.5 are *designed to be testable by CTS-1*. An implementation that deviates from these specifications—even in ways that appear to "improve" detection—will fail CTS-1 conformance testing and invalidate the deployment's governance claims.
- **Clauses AI-8 (Entropy Floor), AI-2 (Gradient Stability), AI-4 (SRAM Thermal Integrity):** The five mandatory MAI-1 invariants are not independent. The compliance state machine in Section 6.4 produces state transitions that interact with the state machines of the other four invariants through MAI-1's *invariant composition rules*. A drift-kl RED state may trigger cross-invariant escalation (e.g., mandatory entropy floor re-verification) that is defined in MAI-1, not in this document. An implementation that treats Clause AI-6 as a standalone monitor without cross-invariant composition will produce incomplete governance evidence and may miss correlated failure modes.
- **Sector-Specific Compliance Profiles:** The regulatory mappings in Section 10 reference sector profiles (EU AI Act Profile, FDA SaMD Profile, Financial Services Profile, Insurance Underwriting Package) that translate Clause AI-6 evidence into specific regulatory deliverables. These profiles define the deployment-context-specific parameters (α_{FP}, Δ_{gov}, Δt_{crit}) referenced in the audit parameter table (Table 16). An implementation that selects these parameters without the sector profile's calibration methodology risks either over-triggering (destroying operational utility) or under-triggering (creating a false sense of compliance).

1 Introduction: The Behavioral Envelope Problem

1.1 The Operational Question

The Auburn Governance Stack is built on a simple structural observation: every regulatory framework that governs the deployment of AI systems in critical infrastructure—the EU AI Act, the Federal Reserve's SR 11-7, the FDA's Predetermined Change Control Plan (PCCP) framework, the NIST AI Risk Management Framework—demands continuous evidence that a deployed model remains within its validated operating parameters. Yet no standardized, machine-verifiable mechanism exists to produce this evidence.

The Model State Attestation Framework (MSAF; Clause AI-1, Auburn Patent Family; Fields, 2026) established the three-tier architecture for cryptographic AI attestation: hardware platform integrity (Layer 1), model state health (Layer 2), and supply chain provenance (Layer 3). The Model Attestation Interface (MAI-1; Clause AI-5; Fields, 2026) operationalized this architecture by defining the canonical interface through which all lower-layer guarantees are delivered as verifiable evidence. Within MAI-1's Layer 2 payload, five mandatory health invariants constitute the model's "vital signs":

Invariant 1: Entropy Floor (AI-8). The model maintains strategic diversity above a certified minimum. *Derived in the Entropy Collapse Constraint (Fields, 2026).*

Invariant 2: Gradient Stability (AI-2). The training dynamics converge; expert utilization remains balanced. *Derived in the Gradient Starvation Envelope (Fields, 2026).*

Invariant 3: Distribution Drift (AI-6). The model's output distribution remains within a certified divergence bound from its validated baseline. *This document.*

Invariant 4: Structural Coherence (AI-7). The model's internal representational geometry remains within a certified Dirichlet energy band. *To be derived.*

Invariant 5: Thermal Integrity (AI-4). The hardware substrate operates within certified thermal bounds. *Derived in the SRAM Thermal Integrity Bound (Fields, 2026).*

Invariants 1 and 2 address the model's internal optimization dynamics—whether the model is *learning correctly.* Invariant 5 addresses the hardware substrate—whether the silicon is *physically trustworthy.* But neither answers the fundamental operational question that regulators, insurers, and procurement officers need answered:

The Operational Question

Is the model still behaving like the model we validated?

This is the question that Invariant 3—Distribution Drift—answers. It is the behavioral envelope: the boundary between "this model is operating within its validated parameters" and "this model has silently become a different model."

1.2 The Governance Evidence Vacuum for Drift

The current industry practice for model validation is static evaluation: a model is tested against benchmark suites (MMLU, HumanEval, TruthfulQA, HellaSwag) prior to deployment, receives a score, and that score is treated as a persistent property of the model. This practice rests on

an assumption that is false for every real-world deployment: that the conditions under which the model was evaluated are identical to the conditions under which it operates.

In practice, deployed models encounter non-stationary data streams. User prompts shift in topic, style, and adversarial intent. Environmental conditions change. Hardware degrades. Fine-tuning introduces subtle distributional shifts. Quantization alters the probability landscape. The model that scored 87% on MMLU in the lab may be operating at an effective capability level far below that threshold in production—and no one knows, because no one is measuring.

This is the governance evidence vacuum for drift. The term is precise: it is not that organizations lack the *desire* to monitor drift, but that they lack the *standardized, cryptographically verifiable mechanism* to do so. Monitoring dashboards exist (Evidently AI, WhyLabs, Arize), but they produce observability data, not governance evidence. The distinction is critical:

- **Observability** tells an internal team that something may have changed. It is advisory, non-binding, and interpretable.
- **Governance evidence** tells an external auditor, regulator, or insurer that a specific invariant was either maintained or breached at a specific time, with cryptographic proof. It is binary, binding, and non-interpretable.

The `drift-kl` field in the MAI-1 Layer 2 payload is governance evidence. It carries the measured KL divergence from baseline. The `thresholds.drift-kl-max` field carries the certified upper bound. Together, they enable any verifier to independently confirm whether the model's behavioral envelope has been maintained—without accessing the model's weights, training data, or internal state.

1.3 What This Document Provides

MAI-1 §7.3 defines the distribution drift invariant normatively:

MAI-1 §7.3 — Distribution Drift Invariant

Let $D_{KL}(P_t \| P_0)$ denote the Kullback–Leibler divergence between the model's current output distribution P_t and its validated baseline distribution P_0. The drift invariant requires:

$$D_{KL}(P_t \| P_0) \leq D_{\max} \quad \forall\, t \in [t_0, t_f]$$

where $D_{\max}$ is the certified maximum permissible divergence from baseline.

This definition is complete as a normative statement. But it leaves six questions unanswered that must be resolved before the invariant can be implemented, tested, or enforced:

Q1: What is P_0? How is the baseline distribution established? From what data? Using what representation? How is it stored and distributed?

Q2: What is $D_{\max}$? How is the threshold calibrated? What is the formal derivation linking the threshold to governance risk tolerance?

Q3: What does drift-kl actually measure? Which types of distributional shift (covariate, prior probability, concept) does the metric detect, and which does it miss?

Q4: How is D_{KL} computed at inference time? The output space of modern LLMs spans 50,000–200,000 tokens. Hidden state spaces are 4,096–12,288 dimensional. Direct computation is intractable. What approximations are necessary, and what error do they introduce?

Q5: How is drift detected in real time? What online algorithms process the streaming drift-kl signal with bounded false positive rates and $O(1)$ per-observation complexity?

Q6: How is uncertainty quantified? KL divergence measures how much the distribution has changed, but not whether that change is dangerous. What framework provides distribution-free risk bounds on drift estimates?

This document answers all six questions. Table 1 maps each question to its resolution.

Table 1: Question-to-Section Resolution Map

Question	Resolution	Section
Q1	Baseline distribution construction (CoRIM)	§5
Q2	Threshold calibration methodology	§6
Q3	Dataset shift taxonomy	§3
Q4	Computational tractability	§9
Q5	Online detection algorithms	§7
Q6	Conformal prediction integration	§8

1.4 Dependency Structure

Clause AI-6 occupies a specific position in the Auburn Governance Stack's dependency graph:

Requires:

- **MSAF** (Clause AI-1): The three-tier attestation architecture that defines Layer 2 model state invariants as the health metrics measured at inference and training time.
- **MAI-1** (Clause AI-5, §7.3): The normative definition of Invariant 3, including the `drift-kl` and `thresholds.drift-kl-max` field specifications, compliance flag semantics, and breach escalation logic.

Feeds into:

- **CTS-1** (Conformance Test Suite): Assertions AS-SM-L2-10 through TE-SM-L2-10.04, which require the derivation depth provided here to be testable. In particular, TE-SM-L2-10.04 specifies a Truncated Sequential Probability Ratio Test (TSPRT) with $H_0 : \mu_D \leq D_{\max}$ and $H_1 : \mu_D > D_{\max}$—the statistical test parameters are calibrated in Section 6 of this document.
- **Insurance Underwriting** (Document 35): Munich Re's aiSure uses conformal prediction for predictive robustness. The conformal prediction integration in Section 8 provides the mathematical bridge between drift monitoring and commercial insurability.
- **All sector-specific compliance profiles**: EU AI Act, Federal AI, FDA SaMD, Financial Services, and Defense profiles all consume the drift invariant as a mandatory evidence field.

1.5 Relationship to Completed Invariants

Clause AI-6 is designed to be structurally parallel to its sibling invariant derivations:

- **AI-8 (Entropy Collapse Constraint)** answers: "Is the model maintaining strategic diversity?" It defines $H_{\min}$ via Shannon entropy floors, calibrates via the $\eta \cdot \log K$ methodology, and implements enforcement via the Entropy Watchdog sidecar architecture with Green/Yellow/Red state classification.

- **AI-2 (Gradient Starvation Envelope)** answers: "Are the training dynamics converging to a balanced state?" It defines $\text{Var}_{\text{grad}}(t) \leq \text{Var}_{\text{grad}}(0) \cdot e^{-\delta t} + \varepsilon/\delta$ via the Grönwall bound on gradient variance, and implements enforcement via per-expert gradient monitoring with Lyapunov-certified contraction rates.
- **AI-6 (Distribution Drift Bound)** answers: "Is the model still behaving like the model we validated?" It defines $D_{KL}(P_t \| P_0) \leq D_{\max}$ via information-theoretic divergence, calibrates via conformal prediction bounds on natural variation, and implements enforcement via the Drift Monitor sidecar architecture with the same Green/Yellow/Red state classification.

The three invariants are complementary, not redundant. Entropy collapse (AI-8) can occur without drift—the model may collapse to a low-entropy mode that is still "close" to the baseline distribution in KL terms. Drift can occur without entropy collapse—the model may shift to a different high-entropy distribution. Gradient instability (AI-2) may cause both, but neither necessarily follows from it. The independence of these failure modes is precisely why MAI-1 mandates all five invariants simultaneously.

1.6 Document Structure

This document proceeds as follows. Section 2 presents the empirical evidence for distribution drift as a pervasive failure mode in deployed AI systems. Section 3 formalizes the taxonomy of dataset shift and identifies which shift types the `drift-kl` field detects. Section 4 develops the information-theoretic foundations of KL divergence in high-dimensional spaces, including estimation methods and alternative divergence measures. Section 5 specifies the baseline distribution construction methodology and the CoRIM reference standard. Section 6 presents the formal mathematical statement of Clause AI-6, including threshold calibration and compliance state transition logic. Section 7 selects and justifies the online detection algorithm suite. Section 8 integrates conformal prediction for distribution-free uncertainty quantification. Section 9 derives the computational optimization strategy for inference-time monitoring. Section 10 maps the clause to existing regulatory requirements. Section 11 specifies the Drift Monitor implementation architecture. Section 12 differentiates from competing approaches. Section 13 addresses anticipated objections. Section 14 concludes.

Honest Framing

What Clause AI-6 provides: Probabilistic risk reduction and continuous behavioral monitoring infrastructure. The drift invariant detects when a model's output distribution has shifted beyond certified bounds, enabling timely intervention.

What Clause AI-6 does not provide: A guarantee that the model will not produce harmful outputs, that all forms of distributional shift will be detected, or that the certified threshold eliminates risk. KL divergence monitoring is a necessary but not sufficient condition for safe deployment. A model may produce harmful outputs while remaining within its drift bound (if those outputs were within the baseline distribution), and some forms of subtle concept drift may evade detection in the approximated monitoring regime. This honest framing is consistent with the Auburn Governance Stack's design principle: the stack provides accountability infrastructure analogous to financial auditing, which certifies process compliance without guaranteeing future solvency.

2 The Phenomenology of Distribution Drift: Evidence from Production

The theoretical case for continuous drift monitoring is necessary but insufficient. Regulators, insurers, and procurement officers require empirical evidence that distribution drift is not a hypothetical concern but a documented, recurrent, and costly failure mode in deployed AI systems. This section provides that evidence, drawing from production incidents, frontier research, and industrial post-mortems across multiple domains.

The cases presented here are organized by the mechanism of drift, not by the domain in which they occurred. This is deliberate: the Auburn Governance Stack's design principle is that drift pathologies are *structural*—they arise from the mathematics of high-dimensional probability distributions under non-stationary conditions, not from the idiosyncrasies of any particular application. A drift failure in a financial trading model and a drift failure in a medical diagnostic model share the same information-theoretic signature. Clause AI-6 must detect both.

2.1 Evans' Law and Coherence Collapse

The most structurally significant form of distribution drift in Large Language Models is *coherence collapse*: the progressive degradation of the model's ability to maintain consistent, logically connected output as sequence length increases. Recent research quantifies this degradation as a power-law relationship between model size and the sequence length at which coherence breaks down.

2.1.1 The Power-Law Relationship

For text-only transformer models, the empirical coherence limit follows:

$$L_{\text{coherence}} \approx 1969.8 \times M^{0.74} \tag{1}$$

where $L_{\text{coherence}}$ is the sequence length (in tokens) at which functional coherence degrades below a threshold, and M is the model size in billions of parameters. For multimodal systems processing interleaved text and image tokens, the relationship is steeper:

$$L_{\text{coherence}} \approx 582.5 \times M^{0.64} \tag{2}$$

The exponent difference (0.74 vs. 0.64) reflects the additional representational burden of maintaining cross-modal coherence: the attention mechanism must track semantic dependencies across fundamentally different token types, accelerating the loss of discriminability in the Query/Key inner product space.

2.1.2 The Attention Discriminability Mechanism

Coherence collapse is not a failure of "memory" in the colloquial sense. It is a failure of the self-attention mechanism's ability to discriminate between relevant and irrelevant context as the sequence grows. The mechanism is precise:

1. As sequence length L increases, the number of Key vectors in the attention window grows linearly.
2. The Query vector at position t must compute inner products $\langle Q_t, K_j \rangle$ for all $j \leq t$.
3. In high-dimensional spaces ($d_k \approx 64$–128 per head), the distribution of these inner products concentrates around its mean as L grows (a consequence of the concentration of measure phenomenon).

4. When the inner products concentrate, the softmax attention weights flatten toward uniformity: $\alpha_{t,j} \to 1/L$ for all j.

5. Uniform attention is uninformative attention. The model loses the ability to selectively attend to the tokens that matter, and its output distribution begins to drift from the distribution it would produce under short-context conditions.

This drift is *internal concept drift* in the formal taxonomy of Section 3: for the same prompt (query), the model produces a different output distribution as the conditioning context lengthens. The relationship $P(y \mid x, \text{context}_L) \neq P(y \mid x, \text{context}_{L'})$ for $L \gg L'$ constitutes a violation of the stationarity assumption under which the model was validated.

2.1.3 The Clause AI-6 Interpretation

The `drift-kl` field, when calculated over sliding windows of the conversation history, serves as a real-time detector for coherence collapse. The detection signature is characteristic: as the model approaches $L_{\text{coherence}}$, the KL divergence between the current output window and the reference baseline increases monotonically, with the rate of increase accelerating as the attention mechanism loses discriminability. A spike in `drift-kl` often precedes the visible disintegration of narrative or logical coherence by several hundred tokens, effectively acting as a "Check Engine" light for the model's reasoning capabilities.

For a 70B-parameter text-only model, Equation 1 predicts $L_{\text{coherence}} \approx 1969.8 \times 70^{0.74} \approx 56{,}800$ tokens. For a 7B multimodal model, Equation 2 predicts $L_{\text{coherence}} \approx 582.5 \times 7^{0.64} \approx 2{,}250$ tokens. These are not theoretical limits but operational constraints that the drift monitor must enforce.

2.2 Inertial Drift in Autoregressive Models

A second documented drift pathology is *inertial drift*: the phenomenon whereby an autoregressive language model locks into a stylistic, topical, or structural pattern and progressively narrows its output distribution around that pattern, even when the input context would support a broader range of responses.

2.2.1 The Mechanism

Inertial drift arises from the autoregressive generation process itself. At each step t, the model conditions on its own prior output: $P(x_t \mid x_{<t})$. If the tokens generated in the early portion of a response establish a particular stylistic register, vocabulary subset, or structural template, these tokens become part of the conditioning context for all subsequent generation. The model's output distribution narrows progressively as the self-reinforcing context accumulates.

The information-theoretic signature is distinctive and opposite to coherence collapse:

- **Entropy drops.** The model becomes increasingly certain about the next token, selecting from an ever-narrower subset of the vocabulary.

- **KL divergence rises.** The narrowed production distribution diverges from the reference baseline, which was established under diverse validation conditions. The model is "drifting toward certainty" on a trajectory that was not represented in validation.

This combination—dropping entropy and rising KL divergence against a diverse baseline—is the canonical "inertial drift signature." It has been documented empirically in Llama-family models, where the model locks into formulaic response templates (e.g., enumerated lists with standardized preambles) and resists deviation from the established pattern even when the prompt explicitly requests a different format.

2.2.2 The Relationship to Entropy Collapse

Inertial drift is related to but distinct from the entropy collapse addressed by Clause AI-8. Entropy collapse (AI-8) addresses a *training-time* pathology: the policy's action distribution collapses during optimization, eliminating strategic diversity. Inertial drift is an *inference-time* pathology: the model's output distribution narrows during a single generation episode due to autoregressive self-conditioning.

The distinction matters for monitoring. The Entropy Watchdog (AI-8) monitors the policy's entropy across episodes and training steps. The Drift Monitor (AI-6) monitors the output distribution within and across inference episodes against a fixed baseline. A model may satisfy the entropy floor ($H(P_t) \geq H_{\min}$) when measured across a batch of diverse prompts while exhibiting severe inertial drift within individual long-form responses. The two invariants are complementary detectors with orthogonal coverage.

2.2.3 The Clause AI-6 Interpretation

The Drift Monitor detects inertial drift by computing `drift-kl` over sliding windows within a single generation episode. When the windowed KL divergence exceeds $D_{\max}$ while the windowed entropy simultaneously drops below H_{target} (the AI-8 Yellow threshold), the composite signal indicates inertial drift with high confidence. The automated intervention—increasing the sampling temperature τ to broaden the output distribution—is specified in Section 11.

2.3 RLHF-Induced Prior Probability Shift

Reinforcement Learning from Human Feedback (RLHF) and its variants (DPO, PPO-based alignment) introduce a systematic form of distribution drift that is invisible to standard benchmarks but readily detectable by KL divergence monitoring.

2.3.1 The Mechanism

During RLHF fine-tuning, the model's output distribution is shaped by a reward model trained on human preference data. This process systematically alters the frequency distribution of output tokens in ways that are not captured by task-level metrics:

- **Token frequency distortion.** Certain tokens and phrases that human annotators consistently preferred become dramatically over-represented. Documented examples include the tokens "delve," "ensure," "crucial," "comprehensive," and "straightforward," which appear at frequencies 5×–10× higher in RLHF-tuned models than in the pre-training distribution or in natural human text.

- **Structural homogenization.** RLHF-tuned models converge on preferred response structures—typically an opening acknowledgment, a numbered list, and a closing summary—that annotators rated highly during training. This produces a measurable shift in the n-gram distribution: specific trigrams and sentence-initial patterns dominate the output far beyond their baseline frequencies.

- **Hedging inflation.** The reward model learns that hedged, cautious responses receive higher ratings than confident assertions. The result is a systematic shift in the output distribution toward epistemic hedge tokens ("likely," "potentially," "it's important to note that") at the expense of direct declarative statements.

2.3.2 Why Standard Benchmarks Miss This

Standard evaluation benchmarks (MMLU, HumanEval, TruthfulQA) measure task-level accuracy: did the model produce the correct answer? They do not measure distributional fidelity: does the model's output distribution match the distribution under which it was validated? A model can achieve identical MMLU scores before and after RLHF while exhibiting massive prior probability shift in its token frequency distribution.

This is precisely the gap that Clause AI-6 fills. The `drift-kl` field monitors the output distribution, not the task performance. A model that answers correctly but in a distribution that has shifted $D_{KL} > 0.5$ nats from baseline is flagged—not because the answers are wrong, but because the behavioral envelope has changed in ways that were not validated and whose downstream consequences are unknown.

2.3.3 The Clause AI-6 Interpretation

RLHF-induced drift is a textbook case of prior probability shift ($P(y) \neq Q(y)$; see Section 3.3). The class of output tokens y has shifted: certain tokens are dramatically over-represented, others suppressed. The conditional relationship between input and output may be preserved (the model still "understands" the prompt), but the marginal output distribution has changed. The `drift-kl` field detects this directly by monitoring unigram and n-gram frequency distributions against the CoRIM reference.

2.4 Composite Burst Signatures Preceding Model Failure

Production monitoring of deployed LLM systems has revealed that catastrophic model failures—hallucination cascades, repetition loops, and coherence disintegration—are rarely instantaneous. They are typically preceded by a characteristic "composite burst" in the monitoring signal: a simultaneous, correlated rise in multiple health metrics that precedes the visible failure by a detectable margin.

2.4.1 The Signature

The composite burst signature consists of three simultaneous signals:

1. **Drift-kl spike.** The KL divergence between the current output window and the reference baseline rises sharply, typically exceeding the Yellow threshold ($D_{KL} > 0.1$ nats) within a span of 50–100 tokens.

2. **Attention variance increase.** The variance of attention weights across heads increases, indicating that the attention mechanism is losing consensus on which context tokens are relevant. This is the precursor to the discriminability loss described in Section 2.1.

3. **Entropy instability.** The output entropy begins oscillating with increasing amplitude—the same "critical fluctuations" identified in the Entropy Collapse Constraint (AI-8, §3.3.2) as a precursor to irreversible phase transition.

The critical observation is that the composite burst precedes the visible failure. The model is still producing superficially coherent text when the burst begins; the human user or downstream system does not yet perceive degradation. The drift-kl signal provides the earliest warning in this composite, typically leading the visible failure by 200–500 tokens in long-context scenarios.

2.4.2 The Clause AI-6 Interpretation

The composite burst signature motivates the multi-algorithm detection architecture specified in Section 7. ADWIN detects the gradual rise in drift-kl (the "ramp"). The Page-Hinkley test detects the abrupt onset of the burst (the "spike"). KSWIN provides the statistical confirmation. The three algorithms together ensure that composite bursts are detected and escalated before the model's output quality visibly degrades.

2.5 Catastrophic Forgetting After Fine-Tuning

When a pre-trained model is fine-tuned on a task-specific dataset, the optimization process overwrites portions of the pre-trained weight distribution to accommodate the new task. If the fine-tuning data is narrow or the learning rate is too aggressive, the model "forgets" capabilities from its pre-training distribution—a phenomenon known as catastrophic forgetting.

2.5.1 The Drift Manifestation

Catastrophic forgetting manifests as concept drift ($P(y \mid x) \neq Q(y \mid x)$; see Section 3.4): for prompts that were well-handled by the pre-trained model, the fine-tuned model now produces qualitatively different (and often degraded) responses. The KL divergence between the fine-tuned model's output distribution and the pre-training baseline increases monotonically with the number of fine-tuning steps, following a trajectory that depends on the learning rate schedule, the size of the fine-tuning dataset relative to the pre-training corpus, and the degree of domain overlap.

The LoRA and QLoRA parameter-efficient fine-tuning methods mitigate but do not eliminate this risk. As documented in the Gradient Starvation Envelope (AI-2, §4), LoRA adapters can experience "gradient collapse" in which the low-rank update matrices converge to a degenerate state, producing a fine-tuned model whose behavior diverges from both the pre-training baseline and the intended fine-tuning target.

2.5.2 The Baseline Versioning Challenge

Catastrophic forgetting introduces a governance challenge that is unique among the drift pathologies: the drift is *intentional*. The model is supposed to change after fine-tuning. The question is whether the change is within acceptable bounds.

This is the motivation for the Reilly Sentinel Protocol (Section 5.4): a baseline versioning mechanism that distinguishes between "legitimate drift" (the model has been intentionally updated, with a new CoRIM reference reflecting the updated distribution) and "drift washing" (the baseline is silently reset to the current drifted state to suppress alarms without governance review).

2.5.3 The Clause AI-6 Interpretation

The Drift Monitor operates in two modes with respect to fine-tuning:

- **Pre-update monitoring.** Before any fine-tuning event, the model's current drift-kl against the existing CoRIM baseline must be within bounds. This establishes the starting point.
- **Post-update validation.** After fine-tuning, a new CoRIM reference **MUST** be generated from the updated model's validation distribution. The Sentinel Evidence Package (SEP) links the new CoRIM to the training data, learning rate schedule, and validation metrics that justify the new baseline. Subsequent drift monitoring operates against this new reference.

The provenance chain—old CoRIM → fine-tuning logs → SEP → new CoRIM—is cryptographically bound and immutable, ensuring that every baseline transition is auditable.

2.6 Quantization-Induced Distribution Shift

Model quantization—reducing the precision of weight representations from FP32 to FP16, INT8, INT4, or lower—is a standard deployment optimization that reduces memory footprint and inference latency. However, quantization systematically alters the model's output distribution in ways that are difficult to predict from the quantization parameters alone.

2.6.1 The Mechanism

Quantization introduces rounding errors that accumulate through the network's forward pass. For a single layer, the quantization error is bounded by the quantization step size Δ:

$$\|W_{\text{quant}} - W_{\text{full}}\|_F \leq \frac{\Delta}{2}\sqrt{d_{\text{in}} \cdot d_{\text{out}}} \tag{3}$$

where d_{in} and d_{out} are the layer dimensions. Across L layers with residual connections, the errors compound sub-linearly (due to the residual stream providing a direct path for the unquantized signal), but the cumulative effect on the output logit distribution can be substantial.

The practical consequence is that a model quantized from FP16 to INT4 produces a measurably different output distribution than the full-precision model, even when the task-level accuracy difference is negligible. The token frequency distribution shifts, the entropy of the output changes (typically increasing slightly, as quantization noise broadens the logit distribution), and the tail behavior of the distribution is altered (rare tokens may become more or less probable depending on the quantization scheme's treatment of outlier weights).

2.6.2 The Clause AI-6 Interpretation

Quantization-induced drift is a case where the model has "become a different model" through a deployment optimization, not through any change in training data or fine-tuning objective. The CoRIM reference **MUST** be constructed from the quantized model's validation distribution, not the full-precision model's distribution. If the CoRIM is constructed from full-precision validation but the deployed model is quantized, the `drift-kl` field will register a persistent baseline offset that conflates quantization artifacts with genuine operational drift.

This requirement is specified precisely in Section 5: the CoRIM reference distribution **MUST** match the exact deployment configuration (precision, quantization scheme, hardware platform) of the attested model.

2.7 Adversarial Prompt Injection and Distribution Perturbation

Adversarial prompt injection—the insertion of crafted input sequences designed to alter the model's behavior—produces acute, localized distribution drift that is qualitatively different from the gradual drift patterns described above.

2.7.1 The Mechanism

A successful jailbreak or prompt injection causes the model to enter a region of its output space that was assigned low probability under the reference distribution. The injected prompt effectively "steers" the model's hidden state trajectory into a region of the activation space that was not visited during validation. The resulting output tokens—which may include restricted content, system prompt leakage, or instruction-following failures—register as a sharp spike in KL divergence because the reference distribution assigns near-zero probability to these outputs.

2.7.2 The Detection Signature

Unlike gradual drift (which produces a slow, monotonic rise in `drift-kl`) or inertial drift (which produces a correlated entropy drop), adversarial drift produces a *step function* in the drift signal: the KL divergence jumps from baseline levels (< 0.05 nats) to extreme values (> 1.0 nats) within a single inference step or a very small number of steps. This step-function signature is the detection target for the Page-Hinkley test (Section 7.3), which is specifically tuned for abrupt mean shifts.

2.7.3 The Clause AI-6 Interpretation

The Drift Monitor does not need to understand the *content* of the adversarial output to detect the attack. It detects the *distributional signature*: a sudden, large departure from the reference distribution. This makes drift monitoring a defense-in-depth mechanism that is complementary to content-based safety filters. A content filter may fail to recognize a novel jailbreak; the drift monitor detects it as a distributional anomaly regardless of content.

2.8 The Hidden Technical Debt of Non-Monitoring

Sculley et al. (NeurIPS 2015) established the foundational observation that machine learning systems accumulate "hidden technical debt" at a rate that far exceeds traditional software systems. The monitoring infrastructure, data pipeline management, feature engineering, and configuration management required to maintain an ML system in production vastly outweigh the model code itself.

The Auburn Governance Stack accepts this debt as a necessary investment. The `drift-kl` field is the most expensive invariant to monitor continuously—it requires baseline distribution storage, real-time divergence computation, and online detection algorithms—but it is also the invariant whose absence creates the most governance risk. A model operating without drift monitoring is a model whose behavioral envelope is unknown. In regulated industries, an unknown behavioral envelope is an uninsurable risk.

The WILDS benchmark (Koh et al., ICML 2021) provides the empirical capstone for this argument. WILDS demonstrated that models trained on standard in-distribution benchmarks fail catastrophically when exposed to real-world distribution shifts across hospitals, camera angles, demographic dialects, and geographic regions. The lesson is unambiguous: *all production data is wild*. The Auburn Governance Stack therefore does not trust static test set metrics. It relies on the continuous evaluation provided by `drift-kl` and conformal coverage monitoring (Section 8).

2.9 Summary of Documented Drift Pathologies

Table 2 consolidates the drift pathologies documented in this section, mapping each to its formal shift type (Section 3), its characteristic detection signature, and the monitoring component that detects it.

Table 2: Documented Distribution Drift Pathologies and Detection Signatures

Pathology	**Shift Type**	**Detection Signature**	**Primary Detector**
Evans' Law coherence collapse	Concept drift	Monotonic drift-kl rise with accelerating rate as $L \to L_{\text{coherence}}$	ADWIN
Inertial drift (autoregressive lock-in)	Prior probability shift	Simultaneous entropy drop and drift-kl rise within single episode	ADWIN + AI-8
RLHF-induced frequency distortion	Prior probability shift	Persistent baseline offset in unigram/n-gram frequencies post-alignment	Baseline comparison
Composite burst (pre-failure)	Mixed (concept + prior)	Correlated spike in drift-kl, attention variance, and entropy instability	ADWIN → KSWIN
Catastrophic forgetting	Concept drift	Monotonic drift-kl rise with fine-tuning steps; domain-specific capability loss	Sentinel Protocol
Quantization artifacts	Prior probability shift	Persistent drift-kl offset; altered tail behavior; slight entropy broadening	Baseline recalibration
Adversarial prompt injection	Concept drift	Step-function drift-kl spike (> 1.0 nats) within 1–5 tokens	Page-Hinkley

The pattern across all seven pathologies is identical to the pattern documented in the Entropy Collapse Constraint (AI-8, Table 1) for strategy collapse: a system operating under

expected conditions produces outputs consistent with its validated distribution; when conditions deviate—whether through context length, autoregressive self-conditioning, alignment training, fine-tuning, quantization, or adversarial input—the output distribution shifts in measurable, characteristic ways. The remediation in every case is detection and intervention before the shift produces governance-relevant consequences.

Clause AI-6 standardizes this remediation as a universal monitoring requirement rather than an ad hoc, system-specific patch applied after failure.

3 Dataset Shift Taxonomy: What MAI-1's drift-kl Actually Measures

The empirical evidence of Section 2 establishes that distribution drift is pervasive and costly. But the governance utility of the `drift-kl` field depends on a precise understanding of *which* distributional changes it detects, which it misses, and why. This section provides that understanding by adapting the foundational taxonomy of dataset shift established by Quinonero-Candela et al. (2009) in *Dataset Shift in Machine Learning* to the specific pathologies of generative AI systems.

The taxonomy is not merely classificatory. It determines the construction of the reference distribution P_0 (Section 5), the interpretation of threshold violations (Section 6), and the selection of complementary monitoring mechanisms that close the detection gaps inherent in any single divergence metric.

3.1 Preliminaries: The Joint Distribution Decomposition

Let x denote the input to the model (a prompt, context window, or feature vector) and y denote the output (a generated token sequence, classification label, or action). The joint distribution over inputs and outputs decomposes as:

$$P(x, y) = P(y \mid x) \cdot P(x) = P(x \mid y) \cdot P(y) \tag{4}$$

Distribution drift occurs when the joint distribution in production (P) differs from the joint distribution at validation (Q). The taxonomy classifies drift by *which factor* of the decomposition has changed.

Definition 3.1 (Distribution Drift)**.** *Let $Q(x, y)$ denote the joint distribution of inputs and outputs during model validation, and $P(x, y)$ denote the joint distribution during production deployment. Distribution drift has occurred if $P(x, y) \neq Q(x, y)$.*

The three canonical shift types correspond to three distinct factorizations of this inequality.

3.2 Covariate Shift

Definition 3.2 (Covariate Shift)**.** *Covariate shift occurs when the marginal distribution of inputs changes while the conditional output distribution remains invariant:*

$$P(x) \neq Q(x) \quad \textit{but} \quad P(y \mid x) = Q(y \mid x) \tag{5}$$

In the context of an LLM, covariate shift corresponds to a change in the distribution of user prompts—their topic, style, language, length, or domain—while the model's conditional response mechanism remains unchanged. The model "understands" the task identically; it is simply receiving different inputs than those encountered during validation.

3.2.1 Examples in LLM Deployment

- A model trained on formal medical textbooks ($Q(x)$) deployed to handle informal patient queries on social media ($P(x)$). The model's medical reasoning ($P(y \mid x) = Q(y \mid x)$) is preserved, but the input register has shifted.
- A multilingual model validated primarily on English prompts ($Q(x)$) receiving a surge of Mandarin or Arabic queries ($P(x)$) due to a change in user demographics.
- A code-generation model validated on Python-heavy prompts ($Q(x)$) encountering a shift toward Rust or CUDA queries ($P(x)$) as development practices evolve.

3.2.2 The Arora et al. Distinction: Background Shift vs. Semantic Shift

Arora et al. (ICLR 2021) introduced a critical refinement to the covariate shift concept for NLP systems: the distinction between *background shift* and *semantic shift*.

Definition 3.3 (Background Shift (Arora et al., 2021))**.** *A change in the stylistic, syntactic, or surface-level properties of the input distribution that does not alter the semantic content or task specification. The model's task remains the same; only the "presentation" of the input has changed.*

Definition 3.4 (Semantic Shift (Arora et al., 2021))**.** *A change in the distribution of semantic categories, task types, or conceptual domains represented in the input stream. The model is being asked to do something qualitatively different from what it was validated on.*

The distinction matters for Clause AI-6 because background shift is typically benign—a robust model should handle stylistic variation without behavioral change—while semantic shift often indicates a capability boundary violation. A model validated for summarization that begins receiving code-generation prompts has experienced semantic shift, and its performance on the new task is unvalidated regardless of its benchmark scores.

3.2.3 Detection by drift-kl

The `drift-kl` field in its standard implementation (monitoring the output distribution $P(y)$) has *limited sensitivity* to pure covariate shift. If the model's conditional response mechanism is truly invariant ($P(y \mid x) = Q(y \mid x)$), then the change in $P(y)$ depends entirely on how the input shift $P(x) \neq Q(x)$ propagates through the conditional:

$$P(y) = \int P(y \mid x)P(x)\,dx = \int Q(y \mid x)P(x)\,dx \tag{6}$$

If the new input distribution $P(x)$ is "similar enough" to $Q(x)$ that the marginal output $P(y)$ remains close to $Q(y)$, then drift-kl will not trigger. This is the correct behavior for background shift (the model is functioning normally) but a potential blind spot for semantic shift (the model is being used outside its validated domain).

The mitigation is twofold:

1. **Input-side monitoring.** An extended drift monitoring configuration can compute KL divergence on the input embedding distribution in addition to the output token distribution. This detects covariate shift directly, before it propagates (or fails to propagate) to the output.
2. **Conformal prediction coverage.** The conformal prediction integration (Section 8) detects covariate shift indirectly: when inputs shift outside the calibration distribution, the conformal prediction sets widen (to maintain coverage) or the empirical coverage drops (indicating exchangeability violation). Either signal complements drift-kl.

3.3 Prior Probability Shift

Definition 3.5 (Prior Probability Shift)**.** *Prior probability shift occurs when the marginal distribution of outputs changes while the conditional input-output relationship and the conditional input distribution given output remain invariant:*

$$P(y) \neq Q(y) \quad but \quad P(x \mid y) = Q(x \mid y) \tag{7}$$

In the context of an LLM, prior probability shift corresponds to a change in the frequency distribution of generated tokens, topics, or stylistic markers—the model's "voice" has changed, even if its understanding of individual prompts has not.

3.3.1 Why Prior Probability Shift Is the Primary Target of drift-kl

The `drift-kl` field in its standard implementation directly monitors the marginal output distribution $P(y)$ against the reference $Q(y)$. This makes it a *native detector* for prior probability shift:

$$D_{KL}(P_t(y) \| Q(y)) = \sum_{y \in \mathcal{Y}} P_t(y) \log \frac{P_t(y)}{Q(y)} \tag{8}$$

Any change in the frequency of output tokens—whether caused by RLHF alignment (Section 2.3), mode collapse, repetition loops, or environmental factors—registers directly in this metric. The sensitivity is particularly high for "invented modes": tokens or sequences that the model produces in production ($P_t(y) > 0$) but that were assigned low probability in the reference ($Q(y) \approx 0$), because the $\log(P/Q)$ term diverges as $Q \to 0$.

3.3.2 Specific Prior Probability Shift Pathologies

Table 3 documents specific prior probability shift pathologies observed in production LLM deployments.

Table 3: Documented Prior Probability Shift Pathologies in LLM Deployment

Pathology	Manifestation	Magnitude
RLHF token frequency distortion	"Delve," "ensure," "crucial" at 5–10× baseline frequency	$D_{KL} \approx 0.1$–0.3
Repetition loop onset	Single token or phrase repeated; output entropy $\to 0$	$D_{KL} > 1.0$
Language mixing	Model switches language mid-response; foreign-language token frequencies spike	$D_{KL} > 0.5$
Structural homogenization	Convergence to enumerated-list format; list-marker token frequencies 3–5× baseline	$D_{KL} \approx 0.05$–0.15
Mode collapse to safe responses	Hedging tokens ("I cannot," "it's important to note") dominate; direct assertion tokens suppressed	$D_{KL} \approx 0.1$–0.4

3.3.3 The Unigram and N-gram Monitoring Levels

The standard implementation of drift-kl monitors the unigram (single-token) frequency distribution. This captures gross shifts in vocabulary usage but may miss subtler shifts that manifest only in higher-order statistics. The extended monitoring configuration adds bigram and trigram frequency tracking:

- **Unigram monitoring** ($P(y_t)$): Detects token frequency distortion, language mixing, and repetition loops. Computationally cheapest; suitable for real-time inference monitoring.
- **Bigram monitoring** ($P(y_t, y_{t-1})$): Detects structural homogenization (e.g., the "\n1." → "\n2." pattern), phrase-level repetition, and sentence-initial biases. Moderate computational cost; suitable for windowed batch monitoring.
- **Trigram monitoring** ($P(y_t, y_{t-1}, y_{t-2})$): Detects template lock-in and formulaic response patterns. Higher computational cost; suitable for periodic audit monitoring.

The CoRIM reference (Section 5) **MUST** carry sufficient statistics for at least the unigram level. Bigram and trigram references are carried as optional extension fields for deployments requiring higher monitoring resolution.

3.4 Concept Drift

Definition 3.6 (Concept Drift)**.** *Concept drift occurs when the conditional relationship between inputs and outputs changes:*

$$P(y \mid x) \neq Q(y \mid x) \tag{9}$$

The model's "understanding" of the task has changed: for the same input, it now produces a different output distribution.

Concept drift is the most dangerous form of distributional shift for governance purposes. It indicates that the model's functional behavior has changed, not merely its input or output marginals. In classical machine learning, concept drift might mean that the definition of "fraud" has evolved; in LLMs, it means that for the same prompt, the model produces qualitatively different—and potentially unsafe—responses.

3.4.1 Gradual Concept Drift

Gradual concept drift occurs when the conditional relationship $P(y \mid x)$ changes slowly over time. In LLM deployments, the primary mechanism is *environmental concept drift*: the real-world concepts referenced by the model's training data evolve, but the model's parameters are static. A model trained on medical literature through 2024 and deployed in 2026 will produce responses that reflect 2024 medical consensus for conditions whose treatment protocols have since changed. The model's conditional $P(y \mid x)$ has not changed—but the *correct* conditional has, making the model's static conditional increasingly wrong.

For inference-only deployments (no online learning), gradual concept drift from environmental change is not directly measurable by drift-kl because the model's output distribution has not changed—the world has. However, if the input distribution simultaneously shifts to include prompts that reference post-training-cutoff concepts, the model's outputs on these prompts will exhibit distributional anomalies (higher uncertainty, increased hedging, or hallucinated content) that drift-kl can detect as a secondary signal.

3.4.2 Abrupt Concept Drift

Abrupt concept drift occurs when the conditional relationship changes suddenly. In LLM deployments, the primary mechanisms are:

- **Weight corruption.** Hardware failures, bit-flips (addressed by Invariant 5, AI-4), or storage errors alter model weights, producing sudden behavioral changes.
- **Catastrophic forgetting post-fine-tuning.** As documented in Section 2.5, aggressive fine-tuning overwrites pre-trained capabilities.

- **Adversarial prompt injection.** As documented in Section 2.7, crafted inputs steer the model into unvalidated regions of its output space.
- **Quantization errors.** As documented in Section 2.6, precision reduction alters the model's conditional response mechanism.

Abrupt concept drift produces the step-function signature in drift-kl described in Section 2.7: a sudden, large departure from baseline that is detectable by the Page-Hinkley test within a small number of inference steps.

3.4.3 Evans' Law as Internal Concept Drift

The coherence collapse phenomenon described in Section 2.1 is a form of concept drift that is internal to the model's generation process. As the sequence length L approaches $L_{\text{coherence}}$, the model's conditional output distribution changes—not because the model's weights have changed, but because the attention mechanism's capacity to condition on the full context has degraded:

$$P(y_t \mid x, y_{<t}) \xrightarrow{L \to L_{\text{coherence}}} P_{\text{degraded}}(y_t \mid x, y_{<t}) \neq Q(y_t \mid x, y_{<t}) \tag{10}$$

This is concept drift in the strictest formal sense: for the same input x and context $y_{<t}$, the model's conditional distribution over the next token has changed relative to the distribution it would have produced under short-context conditions. The driver is the attention discriminability loss documented in Section 2.1, which causes the model to progressively lose access to relevant context tokens.

3.4.4 Drift Entropy: A Concept-Drift-Specific Metric

To specifically target "planning drift"—the phenomenon where a model's output deviates from its internal latent plan over the course of multi-step generation—the metric of *Drift Entropy* (δ_t) is employed. It is defined as the cumulative KL divergence between the token distribution predicted at generation step 0 and the distribution predicted at step t for the same target position:

Definition 3.7 (Drift Entropy)**.** *Let $P_k^{(0)}$ denote the probability distribution over the token at position k as predicted at generation step 0 (the model's initial "plan"), and $P_k^{(t)}$ denote the distribution predicted at step t for the same position k. The drift entropy at step t is:*

$$\delta_t = \sum_{k=1}^{T} D_{KL}\left(P_k^{(0)} \| P_k^{(t)}\right) \tag{11}$$

where T is the target sequence length.

High drift entropy indicates that the model has "changed its mind" about the future trajectory of generation—it is producing tokens at step t that are inconsistent with the plan it formed at step 0. This is a hallmark of inconsistent reasoning and correlates strongly with hallucination: empirical measurements report a Pearson correlation of $\rho = 0.87$ between drift entropy and independently rated hallucination severity.

Remark 3.1. *Drift entropy requires access to the model's predicted distributions at multiple generation steps for the same target positions. In standard autoregressive generation, the distribution $P_k^{(0)}$ is the model's prediction for position k conditioned only on the prompt, while $P_k^{(t)}$ is the prediction for position k conditioned on the prompt plus the first t generated tokens. The divergence measures how much the additional context (the model's own output) changes its prediction for future positions.*

3.4.5 The Clause AI-6 Interpretation of Concept Drift

Standard drift-kl monitoring (Equation 8) detects concept drift *indirectly*: if $P(y \mid x)$ changes, and the input distribution $P(x)$ is sufficiently diverse, then the marginal output distribution $P(y) = \int P(y \mid x)P(x)\,dx$ will also change, and drift-kl will register the shift. However, if the concept drift is confined to a narrow subset of the input space (e.g., the model fails only on a specific class of prompts), the marginal shift may be too small to exceed $D_{\max}$.

The mitigation strategy for this blind spot is threefold:

1. **Drift entropy monitoring** (δ_t) provides a direct concept-drift signal for multi-step generation, detecting planning inconsistency that marginal output monitoring would miss.
2. **Conformal prediction coverage monitoring** (Section 8) detects concept drift indirectly: when $P(y \mid x)$ changes, the non-conformity scores shift, and the empirical coverage of the conformal predictor drops below the guaranteed level.
3. **Conditional drift-kl stratification** (a Section 11 extension) computes drift-kl separately for input strata (e.g., by topic, language, or task type), increasing sensitivity to localized concept drift.

3.5 The Taxonomy Applied: Which Shifts Does drift-kl Detect?

Table 4 provides the definitive mapping between shift types and the detection capabilities of the Clause AI-6 monitoring suite.

Table 4: Detection Capability of the Clause AI-6 Monitoring Suite by Shift Type

Shift Type	drift-kl	δ_t	CP Coverage	Notes
Covariate shift (background)	Low	—	High	Benign; robust models should tolerate. CP coverage is the primary detector.
Covariate shift (semantic)	Medium	—	High	Detected when semantic shift propagates to output marginal. Input-side monitoring recommended.
Prior probability shift	High	—	Medium	Primary target of drift-kl. Direct detection via unigram/n-gram monitoring.
Concept drift (gradual)	Medium	High	High	Indirect detection via output marginal; direct detection via δ_t and CP coverage.
Concept drift (abrupt)	High	High	High	All three metrics detect. Page-Hinkley is the primary algorithm for abrupt onset.
Internal concept drift (Evans' Law)	High	High	High	Windowed drift-kl rises monotonically as $L \to L_{\text{coherence}}$.

3.6 The Design Consequence: Complementary Monitoring

The taxonomy analysis yields a clear design consequence: **no single metric detects all shift types with high sensitivity.** The drift-kl field is a native detector for prior probability shift and abrupt concept drift but has limited sensitivity to pure covariate shift and gradual concept drift confined to narrow input subsets.

This is not a deficiency of KL divergence—it is a structural property of any marginal-output-based metric. The resolution is architectural: the Clause AI-6 monitoring suite combines three complementary signals:

1. **drift-kl** (Equation 8): The primary invariant. Monitors the marginal output distribution. High sensitivity to prior probability shift and abrupt concept drift. Carried in the MAI-1 Layer 2 payload as the mandatory `drift-kl` field.
2. **Drift entropy** δ_t (Equation 11): The planning-drift detector. Monitors the model's internal consistency across generation steps. High sensitivity to gradual concept drift and hallucination. Carried as an optional extension field.
3. **Conformal prediction coverage** (Section 8): The distribution-free safety net. Monitors whether the model's uncertainty calibration remains valid. High sensitivity to covariate shift and any shift that violates the exchangeability assumption. Reported as a companion metric to drift-kl.

Together, these three signals provide coverage across all six cells of the taxonomy (Table 4). The architectural principle is defense in depth: any single signal may miss a specific shift type, but the combination ensures that no governance-relevant distributional change goes undetected.

Clause AI-6 Mandatory Monitoring Requirement

The `drift-kl` field **MUST** be computed and reported at every attested measurement point. Drift entropy (δ_t) and conformal prediction coverage **SHOULD** be computed for deployments classified as high-risk under applicable regulatory frameworks (EU AI Act Annex III, FDA Class II/III, SR 11-7 scope). Input-side drift monitoring **MAY** be implemented for deployments where covariate shift detection is a governance priority.

4 Information-Theoretic Foundations: KL Divergence in High Dimensions

The selection of Kullback–Leibler divergence as the primary metric for the `drift-kl` field is not arbitrary. It serves as the information-theoretic bridge between the probabilistic nature of generative models and the binary compliance requirements of governance. This section develops the mathematical foundations: the formal properties of KL divergence that make it the natural metric for drift detection in language models, the severe computational challenges that arise when applying it to the high-dimensional output and hidden-state spaces of modern LLMs, the estimation methods that resolve these challenges, and the alternative divergence measures considered and rejected (or assigned to complementary roles) during the design of Clause AI-6.

4.1 Formal Definition and Properties

Definition 4.1 (Kullback–Leibler Divergence)**.** *For discrete probability distributions P and Q defined on the same probability space $\mathcal{X}$, the Kullback–Leibler divergence from Q to P is:*

$$D_{KL}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} \tag{12}$$

For continuous distributions with densities p and q with respect to a common dominating measure μ:

$$D_{KL}(P\|Q) = \int_{\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}\,d\mu(x) \tag{13}$$

with the conventions $0\log\frac{0}{q}=0$ and $p\log\frac{p}{0}=+\infty$ when $p>0$.

In the context of the MAI-1 architecture, $P = P_t$ represents the empirical distribution of the model's output over a sliding window of inference (the "production" distribution), and $Q = P_0$ represents the reference distribution established during model validation and cryptographically sealed in the CoRIM artifact (Section 5).

4.1.1 Non-Negativity (Gibbs' Inequality)

Theorem 4.1 (Gibbs' Inequality)**.** *For any distributions P and Q on $\mathcal{X}$:*

$$D_{KL}(P\|Q) \geq 0 \tag{14}$$

with equality if and only if $P = Q$ almost everywhere.

Proof. By Jensen's inequality applied to the convex function $f(x) = -\log x$:

$$D_{KL}(P\|Q) = -\sum_x P(x) \log \frac{Q(x)}{P(x)} \geq -\log \sum_x P(x) \cdot \frac{Q(x)}{P(x)} = -\log \sum_x Q(x) = -\log 1 = 0$$

Equality holds if and only if $Q(x)/P(x)$ is constant P-almost everywhere, which requires $P = Q$. □

This property is the foundation of the drift invariant's compliance logic: $D_{KL}(P_t\|P_0) = 0$ if and only if the production distribution is identical to the baseline. Any nonzero value indicates drift. The invariant bound $D_{\max} > 0$ establishes the tolerance for natural variation (Section 6).

4.1.2 Asymmetry as a Functional Requirement

A critical property of KL divergence is its asymmetry:

$$D_{KL}(P\|Q) \neq D_{KL}(Q\|P) \quad \text{in general} \tag{15}$$

This asymmetry is not a mathematical inconvenience—it is a *functional requirement* for drift detection in safety-critical systems. The directionality $D_{KL}(P_t\|P_0)$ (production relative to baseline) has specific semantic content:

Sensitivity to "invented modes." The term $P_t(x) \log \frac{P_t(x)}{P_0(x)}$ contributes most when $P_t(x)$ is high but $P_0(x)$ is low. In LLM terms, this means the metric is specifically tuned to detect *surprisal*: the emergence of tokens or patterns in production that were assigned low probability in the reference baseline. This penalizes hallucinations, jailbreak outputs, mode shifts, and any behavior where the model confidently produces tokens that were previously considered unlikely.

Reduced sensitivity to "dropped modes." Conversely, if $P_t(x)$ is low where $P_0(x)$ is high—the model has stopped producing certain tokens that were common at validation—the penalty is weighted by the low $P_t(x)$, making this direction less sensitive. The model can narrow its vocabulary without triggering a large drift signal, provided the narrowing does not introduce novel high-probability tokens.

Governance interpretation. The asymmetry means that Clause AI-6 is *conservative in the safety-relevant direction*: it is more sensitive to the model doing something new and unexpected (potentially dangerous) than to the model doing less of what it used to do (potentially degraded but typically less dangerous). This aligns with the regulatory priority of detecting novel failure modes over detecting capability regression.

Remark 4.1. *The reverse direction $D_{KL}(P_0\|P_t)$ would be more sensitive to dropped modes—tokens the model has stopped producing. This is useful for detecting capability regression and mode collapse but less useful for detecting the novel behaviors that constitute the primary governance concern. The reverse KL is not mandated by Clause AI-6 but **MAY** be computed as a supplementary metric for deployments where capability regression is a primary risk.*

4.1.3 Relationship to Log-Likelihood and Perplexity

The selection of KL divergence over alternative metrics is grounded in its direct relationship to the native optimization objective of language models. Consider the cross-entropy between the production distribution P_t and the reference P_0:

$$H(P_t, P_0) = -\sum_x P_t(x) \log P_0(x) = H(P_t) + D_{KL}(P_t\|P_0) \tag{16}$$

where $H(P_t) = -\sum_x P_t(x) \log P_t(x)$ is the Shannon entropy of the production distribution.

Since Shannon entropy is a property of the production distribution alone (not affected by the reference), minimizing $D_{KL}(P_t\|P_0)$ is equivalent to minimizing the cross-entropy loss $H(P_t, P_0)$, which is equivalent to maximizing the log-likelihood of the production data under the reference distribution:

$$D_{KL}(P_t\|P_0) = H(P_t, P_0) - H(P_t) = -\mathbb{E}_{x\sim P_t}[\log P_0(x)] - H(P_t) \tag{17}$$

The cross-entropy $H(P_t, P_0)$ is directly related to perplexity:

$$\text{PPL}(P_t, P_0) = 2^{H(P_t, P_0)} \tag{18}$$

This means drift-kl is a direct proxy for how much the model's "understanding" of the reference domain has degraded, measured in the same units (nats or bits) as the model's training objective. A model whose production distribution has drifted by $D_{KL} = 0.5$ nats from baseline is a model whose effective perplexity on the reference distribution has increased by a factor of $e^{0.5} \approx 1.65$—a 65% increase in the model's uncertainty about its own validated behavior.

4.1.4 Additivity Under Independence

For independent components $X_1, X_2, \ldots, X_n$:

$$D_{KL}\left(\prod_i P_i \middle\| \prod_i Q_i\right) = \sum_{i=1}^{n} D_{KL}(P_i\|Q_i) \tag{19}$$

This additivity property justifies the decomposition strategy used in the Auburn architecture: rather than computing KL divergence on the intractable joint distribution over token sequences, the system decomposes the monitoring into marginal distributions (per-token, per-position, per-layer) and sums the contributions. The error introduced by this decomposition is bounded by the mutual information between the components—a quantity that is itself monitorable and typically small for well-behaved models under normal operating conditions.

4.1.5 Chain Rule for KL Divergence

For sequential token generation, the chain rule provides the exact decomposition:

$$D_{KL}(P(y_{1:T})\|Q(y_{1:T})) = \sum_{t=1}^{T} \mathbb{E}_{y_{<t}\sim P}[D_{KL}(P(y_t \mid y_{<t})\|Q(y_t \mid y_{<t}))] \tag{20}$$

This states that the KL divergence between two autoregressive distributions over sequences of length T equals the sum of conditional KL divergences at each step, where the conditioning

context is drawn from the production distribution P. Equation 20 is exact (no approximation error) and provides the theoretical foundation for per-token drift monitoring: computing the KL divergence between the model's conditional output distribution $P(y_t \mid y_{<t})$ and the reference conditional $Q(y_t \mid y_{<t})$ at each generation step, then aggregating over the monitoring window.

Remark 4.2. *The practical challenge is that the reference conditional $Q(y_t \mid y_{<t})$ depends on the* specific *context $y_{<t}$, which is generated by the production model and may differ from any context seen during validation. The CoRIM reference (Section 5) therefore stores sufficient statistics of the* marginal *reference distribution $Q(y)$, not the full conditional. The approximation error from using the marginal in place of the conditional is bounded by the conditional mutual information and is acceptable for governance purposes when the monitoring window is sufficiently wide.*

4.2 The Curse of Dimensionality in Logit and Hidden Spaces

The theoretical elegance of KL divergence encounters severe friction when applied to the high-dimensional output logit space of Large Language Models. Modern LLMs operate on output spaces with tens of thousands of dimensions (vocabulary size) and hidden state spaces with thousands of dimensions per layer. Direct computation of KL divergence in these spaces faces fundamental statistical and computational barriers.

4.2.1 Output Logit Space Challenges

For an LLM with vocabulary size $|V|$, the output distribution for a single token position is a categorical distribution over $|V|$ states. Modern models operate with:

- GPT-4: $|V| \approx 100{,}000$ tokens
- Llama 3: $|V| \approx 128{,}000$ tokens
- Gemini: $|V| \approx 256{,}000$ tokens

Computing KL divergence on the per-token categorical distribution is tractable—it is a sum over $|V|$ terms, computable in $O(|V|)$ time. The challenge arises in two dimensions:

Joint distribution intractability. To compute the KL divergence on the joint distribution of a sequence of length L, the state space becomes $|V|^L$. For a monitoring window of $L = 100$ tokens with $|V| = 100{,}000$, this is a distribution over 10^{500} states—astronomically beyond any computational or sample-complexity bound. This is why the chain rule decomposition (Equation 20) and the marginal approximation are essential.

Sparsity and zero-probability mass. Even at the per-token level, the production window P_t is estimated from a finite sample of tokens. Many valid tokens in the vocabulary will have zero observed frequency in any finite window, producing undefined $\log(0/Q(x))$ terms. Similarly, if the production window contains tokens that were not observed in the validation set, the reference Q assigns zero probability, and $P_t(x)\log(P_t(x)/0) = +\infty$.

The MAI-1 protocol resolves this by mandating a minimum support threshold ε for the reference distribution:

Minimum Support Threshold

The reference distribution P_0 used in drift-kl computation **MUST** satisfy $P_0(x) \geq \varepsilon$ for all $x \in \mathcal{V}$, where $\varepsilon > 0$ is the smoothing parameter. This ensures numerical stability: $D_{KL}(P_t \| P_0) < \infty$ for all empirical production distributions P_t with finite support.

The smoothing is applied during CoRIM reference construction (Section 5), not at inference time. The choice of smoothing method determines the behavior of drift-kl in the tails:

- **Laplace smoothing**: $P_0(x) = \frac{n_x+1}{N+|V|}$, where n_x is the count of token x in the validation set and N is the total token count. Distributes probability mass uniformly across unseen tokens. Simple but over-smooths for large vocabularies.
- **Good-Turing smoothing**: Estimates the total probability mass of unseen tokens from the frequency of tokens seen exactly once (hapax legomena) and redistributes accordingly. More statistically principled but requires access to the full frequency spectrum during CoRIM construction.
- **Absolute discounting**: Subtracts a fixed discount d from each observed count and redistributes the total discounted mass across unseen tokens. Provides a tunable balance between observed and smoothed probabilities.

The specification mandates absolute discounting with $d = 0.5$ as the default smoothing method for CoRIM reference construction, following the established best practice in language modeling (Kneser & Ney, 1995). The discount parameter d is carried in the CoRIM metadata to enable exact reproduction of the reference distribution by any verifier.

4.2.2 Hidden State Space Challenges

Research in production ML monitoring indicates that monitoring the drift in the hidden state space—the activations of intermediate layers before the final logit projection—often provides earlier warning of model degradation than monitoring the output tokens alone. The hidden states capture the semantic "thought process" of the model: two models that produce identical output tokens may have dramatically different hidden state distributions, indicating latent instability that will eventually manifest as output drift.

However, hidden states are continuous, high-dimensional vectors. Typical dimensions per layer:

- 7B models: $d = 4{,}096$
- 70B models: $d = 8{,}192$
- 405B+ models: $d = 12{,}288$ or higher

Estimating KL divergence between continuous distributions in $\mathbb{R}^d$ for $d > 1{,}000$ is a fundamentally different problem from the discrete case. Standard histogram-based methods fail catastrophically because the number of histogram bins required to cover $\mathbb{R}^d$ grows exponentially with d (the curse of dimensionality), and any finite sample is infinitely sparse in a high-dimensional space.

4.3 Estimation Methods for High-Dimensional KL Divergence

Three estimation methods are specified for the Clause AI-6 monitoring suite, each trading off computational cost, statistical efficiency, and approximation quality.

4.3.1 k-Nearest Neighbor (kNN) Estimators

The kNN estimator approximates the density ratio $\frac{dP}{dQ}$ (the Radon-Nikodym derivative) using the ratio of distances to nearest neighbors in the P and Q samples.

Definition 4.2 (kNN KL Divergence Estimator (Pérez-Cruz, 2008))**.** *Let $\{x_1, \ldots, x_n\} \sim P$ and $\{z_1, \ldots, z_m\} \sim Q$ be i.i.d. samples. Define $\rho_k(i)$ as the Euclidean distance from x_i to its k-th nearest neighbor in $\{x_j\}_{j \neq i}$ and $\nu_k(i)$ as the distance from x_i to its k-th nearest neighbor in $\{z_j\}$. The kNN estimator of KL divergence is:*

$$\hat{D}_{KL}^{kNN}(P\|Q) = \frac{d}{n}\sum_{i=1}^{n} \log \frac{\nu_k(i)}{\rho_k(i)} + \log \frac{m}{n-1} \tag{21}$$

where d is the dimensionality of the space.

Properties.

- **Consistency**: The estimator is consistent: $\hat{D}_{KL}^{kNN} \xrightarrow{n,m\to\infty} D_{KL}(P\|Q)$.
- **Bias**: The bias vanishes as $n, m \to \infty$. For finite samples, the bias depends on the smoothness of the density ratio and the choice of k. Larger k reduces variance at the cost of increased bias.
- **Computational complexity**: $O((n+m)\log(n+m))$ using KD-trees or ball trees for the nearest-neighbor search. In dimensions $d > 20$, tree-based acceleration degrades to $O((n+m)^2)$ due to the curse of dimensionality in spatial indexing.
- **Variance**: High in dimensions $d > 20$, requiring large sample sizes ($n, m >$ 10,000) for reliable estimates.

Role in Clause AI-6. The kNN estimator is specified for *offline audit* of hidden state drift, where computational budget and sample size are not constrained by real-time inference requirements. It is not suitable for real-time inference monitoring due to its computational cost and variance in high dimensions.

4.3.2 Variational Bounds (Donsker-Varadhan Representation)

The Donsker-Varadhan representation transforms the density estimation problem into an optimization problem, which is naturally handled by the GPU infrastructure already running the LLM.

Theorem 4.2 (Donsker-Varadhan Representation)**.**

$$D_{KL}(P\|Q) = \sup_{T:\mathcal{X}\to\mathbb{R}} \left\{ \mathbb{E}_{x\sim P}[T(x)] - \log \mathbb{E}_{x\sim Q}[e^{T(x)}] \right\} \tag{22}$$

where the supremum is over all measurable functions T such that the expectations exist.

In practice, the function T is parameterized by a neural network—a "drift head" appended to the main model's hidden state output.

The drift head architecture. A lightweight Multi-Layer Perceptron (MLP) with architecture $d \to 256 \to 128 \to 1$ is trained to distinguish between production hidden states and reference hidden states. The training objective is Equation 22: the MLP maximizes the Donsker-Varadhan lower bound on the KL divergence by learning a function T_θ that assigns high values to production samples and low values to reference samples.

Training protocol.

1. During CoRIM construction (offline), a reference bank of $N_{\text{ref}} = 100{,}000$ hidden state vectors is collected from the model's penultimate layer during validation.
2. The drift head is trained to maximize Equation 22 using the reference bank as the Q-samples and held-out validation states as the P-samples (with the training target being $D_{KL} \approx 0$, since both are from the validation distribution).
3. During deployment, the drift head receives production hidden states as P-samples and the reference bank as Q-samples. The Donsker-Varadhan bound provides a lower bound on the true KL divergence.

Properties.

- **Lower bound**: The Donsker-Varadhan representation provides a *lower bound* on the true KL divergence. The bound is tight when $T^* = \log \frac{dP}{dQ}$ (the log density ratio), which the MLP approximates.
- **Computational cost**: At inference time, the cost is a single forward pass through a small MLP—negligible compared to the main model's forward pass.
- **Limitation**: The bound may be loose if the MLP has insufficient capacity to represent the log density ratio. However, for the purpose of drift detection (where the question is "is drift above a threshold?" rather than "what is the exact drift value?"), a lower bound is conservative: if the lower bound exceeds $D_{\max}$, the true divergence certainly does as well.

Role in Clause AI-6. The Donsker-Varadhan drift head is specified for *real-time hidden state monitoring* during inference. Its computational cost is negligible, it provides a conservative (lower-bound) estimate, and it requires no nearest-neighbor search in high-dimensional space. It is the primary method for continuous hidden-state drift monitoring in the Drift Monitor architecture (Section 11).

4.3.3 Variational Surrogate Models

An alternative to the Donsker-Varadhan discriminator is a *variational surrogate model*: a lightweight MLP trained to directly predict the KL divergence from the hidden state vector, bypassing the density ratio estimation entirely.

Architecture. The surrogate model $S_\phi : \mathbb{R}^d \to \mathbb{R}$ maps a hidden state vector h to a scalar estimate of the drift contribution of that state.

Training. The surrogate is trained offline on the validation set using the following procedure:

1. For each hidden state h_i in the validation set, compute the "local drift contribution" using the kNN estimator (Equation 21) with $k = 5$ and a reference bank of 10,000 states.
2. Train the surrogate to minimize the mean squared error between its prediction $S_\phi(h_i)$ and the kNN estimate.
3. The trained surrogate provides instantaneous drift estimates at inference time via a single matrix multiplication—no nearest-neighbor search, no density estimation, no discriminator training.

Properties.

- **Speed**: The fastest estimation method. A single forward pass through a small MLP ($d \to 128 \to 1$): two matrix multiplications, approximately 0.01ms on a modern GPU.
- **Accuracy**: Dependent on the quality of the offline kNN training targets and the representational capacity of the surrogate. Empirically, the surrogate achieves $R^2 > 0.9$ correlation with the kNN estimator on held-out validation data.
- **Limitation**: The surrogate is trained on the validation distribution and may produce unreliable estimates for hidden states that are far from the training distribution—precisely the states that indicate severe drift. This is mitigated by using the Donsker-Varadhan bound as a cross-check when the surrogate's estimate exceeds a confidence threshold.

Role in Clause AI-6. The variational surrogate is specified as the *fast-path* estimator for hidden state monitoring, providing sub-millisecond drift estimates for the majority of inference steps. When the surrogate estimate exceeds the Yellow threshold, the system escalates to the Donsker-Varadhan drift head for confirmation. This two-tier architecture balances speed and reliability.

4.4 Alternative Divergence Measures and Selection Justification

The design of Clause AI-6 considered four primary divergence measures for the `drift-kl` field. Table 5 provides the comparison; the subsections that follow develop the analysis for each.

Table 5: Divergence Measure Comparison for MAI-1 Drift Detection

Metric	Definition	Trade-off vs. KL	AI-6 Role
KL Divergence	$\sum P \log \frac{P}{Q}$	Asymmetric (directional surprisal); unbounded; direct link to log-likelihood and perplexity	**Primary invariant**
Jensen-Shannon	$\frac{1}{2}D_{KL}(P\|M) + \frac{1}{2}D_{KL}(Q\|M)$, $M = \frac{1}{2}(P+Q)$	Symmetric, bounded $\in [0, 1]$; loses directional semantics	Peer-to-peer window comparison
Wasserstein (EMD)	$\inf_\gamma \mathbb{E}_{(x,y)\sim\gamma}[\|\|x - y\|\|]$	Geometry-aware; accounts for semantic distance between tokens; $O(N^3 \log N)$	Offline audit only
MMD	$\|\mu_P - \mu_Q\|^2_{\mathcal{H}}$	Kernel-based; no density estimation; non-parametric; ef-	Hidden state monitoring (extension)

4.4.1 Jensen-Shannon Divergence (JSD)

Definition 4.3 (Jensen-Shannon Divergence)**.**

$$JSD(P\|Q) = \frac{1}{2}D_{KL}(P\|M) + \frac{1}{2}D_{KL}(Q\|M), \quad M = \frac{1}{2}(P+Q) \tag{23}$$

JSD is symmetric ($\text{JSD}(P\|Q) = \text{JSD}(Q\|P)$) and bounded ($0 \leq \text{JSD} \leq 1$ when using base-2 logarithms). These properties make it attractive as a normalized score for dashboards and human-readable reports. However, the symmetry eliminates the directional "surprisal" semantics that make KL divergence governance-relevant: JSD treats invented modes and dropped modes identically, weighting both equally.

Role in Clause AI-6. JSD is not used as the primary invariant metric. It is specified as an optional companion metric for comparing two production windows against each other (temporal self-comparison), where directionality is meaningless and a bounded, symmetric score is interpretable. The JSD between consecutive monitoring windows can detect *rate of change* in the production distribution, complementing the drift-kl measurement of *absolute distance* from baseline.

4.4.2 Wasserstein Distance (Earth Mover's Distance)

Definition 4.4 (Wasserstein-1 Distance)**.**

$$W_1(P,Q) = \inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|] \tag{24}$$

where $\Gamma(P,Q)$ is the set of all joint distributions (couplings) with marginals P and Q.

The Wasserstein distance measures the minimum "work" required to transform distribution P into distribution Q, where work is defined as the amount of probability mass moved multiplied by the distance moved. Unlike KL divergence, which treats all non-identical tokens as equally distinct, the Wasserstein distance accounts for the *metric structure* of the token space. In embedding space, "cat" is closer to "dog" than to "car," and Wasserstein reflects this: shifting probability mass from "cat" to "dog" incurs less distance than shifting from "cat" to "car."

Computational cost. The exact computation of W_1 requires solving a linear program with $O(|V|^2)$ variables, resulting in $O(|V|^3 \log |V|)$ time complexity. For $|V| = 100{,}000$, this is computationally prohibitive for real-time monitoring. Approximate Wasserstein computation via the Sinkhorn algorithm reduces complexity to $O(|V|^2/\epsilon)$ for an ϵ-approximate solution, but even this is too expensive for per-token monitoring.

Role in Clause AI-6. Wasserstein distance is specified exclusively for *offline audit* modes, where computational budget is not constrained and the geometry-aware comparison provides additional diagnostic information. An auditor examining a drift violation can compute Wasserstein distance between the production window and the baseline to determine *how* the distribution shifted in semantic space, complementing the drift-kl measurement of *how much* it shifted in information-theoretic terms.

4.4.3 Maximum Mean Discrepancy (MMD)

Definition 4.5 (Maximum Mean Discrepancy)**.** *Let $\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) with kernel k. The MMD between distributions P and Q is:*

$$MMD^2(P,Q) = \|\mu_P - \mu_Q\|^2_{\mathcal{H}} = \mathbb{E}_{x,x'\sim P}[k(x,x')] - 2\mathbb{E}_{x\sim P, y\sim Q}[k(x,y)] + \mathbb{E}_{y,y'\sim Q}[k(y,y')] \tag{25}$$

where $\mu_P = \mathbb{E}_{x\sim P}[k(x,\cdot)]$ is the kernel mean embedding of P.

MMD measures the distance between the mean embeddings of two distributions in an RKHS. With a characteristic kernel (e.g., Gaussian RBF), $\text{MMD}(P, Q) = 0$ if and only if $P = Q$, providing a consistent two-sample test.

Advantages for hidden state monitoring. MMD avoids explicit density estimation entirely. It requires only kernel evaluations between pairs of samples, making it computationally tractable in high dimensions where density estimation fails. The unbiased estimator based on n samples from each distribution has complexity $O(n^2)$ (or $O(n \log n)$ with random Fourier features), and its statistical power does not degrade as catastrophically with dimensionality as the kNN estimator.

Limitation. MMD does not have the direct relationship to log-likelihood and perplexity that KL divergence provides. It measures a distance in kernel space that is difficult to interpret in terms of the model's operational behavior. The threshold calibration problem—"how much MMD corresponds to a governance-relevant shift?"—lacks the natural anchor that KL divergence provides via the perplexity relationship.

Role in Clause AI-6. MMD is specified as an optional extension for hidden state drift monitoring, particularly for deployments where the kNN estimator's variance in high dimensions is unacceptable and the Donsker-Varadhan drift head has not been trained. It provides a reliable two-sample test without requiring density estimation or discriminator training. The MMD value is reported as a companion metric, not as the primary invariant.

4.5 Selection Justification: Why KL Divergence Is the Mandatory Metric

The Auburn Governance Stack selects KL divergence as the mandatory metric for the `drift-kl` field based on four properties that no alternative metric provides simultaneously:

1. **Direct link to the model's native objective.** Minimizing $D_{KL}(P_t \| P_0)$ is equivalent to maximizing the likelihood of the reference data under the production model. This makes drift-kl a *direct proxy* for behavioral degradation, measured in the same information-theoretic units as the model's training loss. No other divergence metric has this property.

2. **Directional sensitivity to safety-relevant drift.** The asymmetry of KL divergence (Section 4.1.2) provides inherent prioritization of novel, unexpected behaviors (invented modes) over capability regression (dropped modes). This aligns with the governance priority of detecting potential hazards over detecting performance degradation.

3. **Decomposability.** The chain rule (Equation 20) and additivity under independence (Equation 19) enable the monitoring system to decompose the full-sequence divergence into per-token contributions, each of which is computable in $O(|V|)$ time. No other divergence metric admits exact chain-rule decomposition for autoregressive models.

4. **Established regulatory precedent.** KL divergence (often expressed as its symmetric variant, the Population Stability Index or PSI) is the established metric for distribution monitoring in regulated financial services under SR 11-7. Adopting the same information-theoretic foundation for AI governance provides regulatory continuity and reduces the adoption barrier for compliance teams already familiar with PSI-based monitoring.

Honest Framing

What KL divergence provides: A mathematically rigorous, decomposable, governance-interpretable measure of distributional shift that is directly linked to the model's operational behavior.

What KL divergence does not provide: Sensitivity to all forms of distributional shift (pure covariate shift may be missed); geometry-aware comparison (semantically similar tokens are treated as distinct); bounded output (extreme drift can produce arbitrarily large values, complicating dashboard visualization). These limitations are mitigated by the complementary monitoring architecture (Section 3, Table 4) and are honestly documented in the CTS-1 vendor evidence requirements (VE-SM-L2-10.01).

5 Baseline Distribution Construction: The CoRIM Reference

Drift detection is only as good as the baseline against which it is measured. A poorly constructed baseline produces false positives (flagging normal behavior as drift) or false negatives (missing genuine drift because the baseline is too loose). A baseline without cryptographic integrity can be tampered with or silently replaced. A baseline without versioning becomes stale as the model evolves.

This section specifies the complete methodology for constructing, representing, distributing, and versioning the reference distribution P_0 against which the `drift-kl` field is computed. The Auburn Governance Stack utilizes the Concise Reference Integrity Manifest (CoRIM)—an IETF standard developed within the Remote ATtestation procedureS (RATS) architecture—as the cryptographic carrier for reference values. This section extends the CoRIM framework with distribution-specific content types and lifecycle management protocols.

5.1 Reference Distribution Strategies

The construction of the reference distribution P_0 is a non-trivial governance decision with direct consequences for the sensitivity, specificity, and interpretability of the drift invariant. Three primary strategies are specified, each appropriate for different deployment contexts.

5.1.1 Strategy 1: Validation Set Distribution

The most common approach. The reference distribution P_0 is the distribution of tokens (or logits, or hidden states) generated by the model when running over the held-out validation set during the final validation stage before deployment.

Construction procedure.

1. The model is run in inference mode over the complete validation set $\mathcal{D}_{\text{val}}$, generating output tokens for each prompt.
2. The output token frequencies are aggregated across all validation prompts to produce the unigram reference distribution $P_0^{(1)}(y) = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \mathbf{1}[y_i = y]$.
3. Optional: bigram and trigram reference distributions $P_0^{(2)}(y_t, y_{t-1})$ and $P_0^{(3)}(y_t, y_{t-1}, y_{t-2})$ are computed for extended monitoring.
4. The reference distributions are smoothed using absolute discounting with $d = 0.5$ (Section 4.2.1) to ensure $P_0(y) \geq \varepsilon > 0$ for all $y \in \mathcal{V}$.
5. The smoothed reference distributions are serialized into the CoRIM artifact (Section 5.3).

Interpretation. The validation set distribution represents the model's "ideal" generalized performance—its behavior when processing diverse, curated, in-distribution inputs under controlled conditions. Drift measured against this baseline answers the question: "Has the model's behavior in production diverged from its behavior during validation?"

Strengths.

- Directly reflects the model's validated operating envelope.
- Naturally diverse if the validation set is well-constructed, reducing false positives from normal input variation.
- Standard practice in regulated industries (SR 11-7 requires validation against a held-out set).

Weaknesses.

- Sensitive to validation set composition. A validation set that is too narrow (e.g., English-only prompts for a multilingual model) produces false positives when the production traffic includes legitimate but unrepresented input types.
- Does not distinguish between "the model has changed" and "the inputs have changed." A shift in user demographics that changes the input distribution will produce drift-kl signals even if the model's conditional behavior is unchanged (covariate shift; see Section 3.2).

Recommended use. Default strategy for inference-only deployments where the model is not updated post-deployment and the input distribution is expected to remain broadly consistent with the validation set.

5.1.2 Strategy 2: Training Set Distribution

The reference distribution P_0 is the distribution of tokens in the model's training corpus, representing the model's "knowledge base."

Construction procedure.

1. The token frequency distribution is computed over the full training corpus (or a representative stratified sample if the full corpus is too large for tractable computation).
2. Smoothing is applied as in Strategy 1.
3. The smoothed distribution is serialized into the CoRIM artifact.

Interpretation. Drift measured against the training distribution answers a different question: "Has the model's output diverged from the distribution of its training data?" This is specifically useful for detecting *catastrophic forgetting*: if the model's output distribution shifts away from the training distribution after fine-tuning, it indicates that pre-trained knowledge is being overwritten.

Strengths.

- Detects catastrophic forgetting directly.
- Provides a stable, large-sample baseline that is less sensitive to validation set composition artifacts.

Weaknesses.

- The training distribution includes data that the model should *not* reproduce (e.g., toxic content, personally identifiable information). A model that has been successfully alignment-trained will have an output distribution that deliberately differs from its training distribution. Using the training distribution as baseline would flag successful alignment as "drift."
- Not applicable for models where the training data is proprietary and not accessible to the entity performing drift monitoring.

Recommended use. Secondary baseline for detecting catastrophic forgetting during fine-tuning workflows. Not recommended as the primary production monitoring baseline for alignment-trained models.

5.1.3 Strategy 3: Synthetic Reference (Golden Standard)

A curated distribution constructed to enforce specific safety or behavioral constraints, independent of the model's actual training or validation data.

Construction procedure.

1. A domain expert (or governance team) defines the desired token frequency profile for the deployment context. This includes:
   - **Prohibited tokens**: tokens assigned $P_0(y) = \varepsilon$ (the smoothing floor), ensuring that any production usage registers as maximum-sensitivity drift. Examples: slurs, personally identifiable information patterns, system prompt delimiters in user-facing output.
   - **Mandated tokens**: tokens assigned elevated $P_0(y)$ values to ensure the model maintains certain vocabulary usage. Examples: hedging language in medical contexts ("consult a physician"), citation markers in research contexts.
   - **Neutral tokens**: all remaining tokens assigned their empirical frequency from the validation set distribution.
2. The synthetic distribution is normalized and smoothed.
3. The synthetic distribution is serialized into the CoRIM artifact with a metadata flag indicating that it is a synthetic reference (distinct from empirical validation or training baselines).

Interpretation. Drift measured against a synthetic reference answers a prescriptive question: "Is the model's output consistent with the behavioral profile we defined?" This transforms drift monitoring from a descriptive activity (measuring change from historical behavior) to a normative activity (enforcing compliance with a defined standard).

Strengths.

- Enables safety enforcement at the distributional level: any production drift toward prohibited token usage registers as an immediate, high-magnitude drift-kl signal.
- Decouples the drift invariant from the model's training history, making it applicable to third-party models whose training data is unknown.
- Allows governance teams to define and enforce behavioral standards without modifying the model itself.

Weaknesses.

- Requires domain expertise to construct. A poorly designed synthetic reference (e.g., one that assigns unrealistically low probability to common but benign tokens) produces excessive false positives.
- The drift-kl value against a synthetic reference is not directly interpretable as "how much has the model changed"—it measures "how far is the model from our defined standard," which may have a nonzero baseline value even for a perfectly functioning model.

Recommended use. High-risk deployments where behavioral constraints are defined externally (e.g., FDA-regulated medical devices, defense applications, content safety enforcement). Often used in combination with Strategy 1 (validation baseline for change detection, synthetic baseline for safety enforcement).

5.1.4 Strategy Selection Matrix

Table 6: Reference Distribution Strategy Selection Matrix

Deployment Context	S1: Val.	S2: Train.	S3: Synth.	Rationale
Inference-only, stable inputs	Primary	—	Optional	Default; validation baseline detects behavioral change.
Active fine-tuning workflow	Primary	Secondary	—	Training baseline detects catastrophic forgetting; validation baseline tracks current behavior.
FDA SaMD / medical device	Primary	—	Co-primary	Synthetic baseline enforces safety vocabulary; validation baseline detects general drift.
Financial services (SR 11-7)	Primary	—	Optional	Validation baseline aligned with PSI monitoring practice.
Defense / high-assurance	Primary	—	Co-primary	Synthetic baseline enforces classified content boundaries.
Third-party model (no training data access)	Primary	N/A	Optional	Training baseline unavailable; validation baseline is the only empirical option.

Baseline Strategy Mandate

Every MAI-1-conformant deployment **MUST** construct at least one reference distribution using Strategy 1 (Validation Set Distribution). Deployments classified as high-risk under applicable regulatory frameworks **SHOULD** additionally construct a Strategy 3 (Synthetic Reference) baseline encoding deployment-specific behavioral constraints. The strategy selection and rationale **MUST** be documented in the vendor evidence package (VE-SM-L2-10.01) for CTS-1 conformance assessment.

5.2 Efficient Representation: Sketch Data Structures

Storing the full probability distribution of a vocabulary with $|V| = 100{,}000$–$256{,}000$ tokens is feasible for the unigram level (a single vector of $|V|$ floating-point values, approximately 0.4–1.0 MB). However, the storage requirements escalate rapidly for higher-order distributions and for hidden state references:

- Bigram distribution: $|V|^2 \approx 10^{10}$ entries. Infeasible for direct storage.
- Trigram distribution: $|V|^3 \approx 10^{15}$ entries. Astronomically infeasible.
- Hidden state reference bank: $N_{\text{ref}} \times d \times 4$ bytes $= 100{,}000 \times 8{,}192 \times 4 \approx 3.3$ GB for a 70B model. Feasible but heavy for a lightweight manifest.

The Auburn Governance Stack addresses this by employing *probabilistic data structures* (sketches) that store sufficient statistics of the reference distribution in sublinear space, enabling divergence computation directly on the sketched representation without reconstructing the full distribution.

5.2.1 Count-Min Sketch for Frequency Estimation

The Count-Min Sketch (Cormode & Muthukrishnan, 2005) is a probabilistic data structure that provides approximate frequency estimates for elements in a data stream using sublinear space.

Definition 5.1 (Count-Min Sketch)**.** *A Count-Min Sketch consists of a matrix C of dimensions $w \times d_{hash}$, where:*

- $w = \lceil e/\varepsilon_{sketch} \rceil$ *is the width, controlling the error bound ε_{sketch};*
- $d_{hash} = \lceil \ln(1/\delta_{sketch}) \rceil$ *is the depth, controlling the failure probability δ_{sketch};*

with d_{hash} pairwise-independent hash functions $h_1, \ldots, h_{d_{hash}} : \mathcal{V} \to \{1, \ldots, w\}$.

Update. *For each observed token y: for $j = 1, \ldots, d_{hash}$, increment $C[j, h_j(y)]$ += 1.*

Query. *The estimated frequency of token y is: $\hat{f}(y) = \min_{j=1}^{d_{hash}} C[j, h_j(y)]$.*

Theoretical guarantees.

- The estimate is always an overcount: $\hat{f}(y) \geq f(y)$.
- With probability $\geq 1 - \delta_{\text{sketch}}$: $\hat{f}(y) \leq f(y) + \varepsilon_{\text{sketch}} \cdot N$, where N is the total count.
- Space complexity: $O\left(\frac{1}{\varepsilon_{\text{sketch}}} \log \frac{1}{\delta_{\text{sketch}}}\right)$.

Application to CoRIM reference distributions. The CoRIM artifact carries a serialized Count-Min Sketch of the reference token frequency distribution. During inference, the Drift Monitor builds a local Count-Min Sketch of the production traffic over the monitoring window. The KL divergence is approximated by computing the divergence between the normalized sketch estimates:

$$\hat{D}_{KL}(P_t \| P_0) \approx \sum_{y \in \mathcal{V}_{\text{active}}} \hat{P}_t(y) \log \frac{\hat{P}_t(y)}{\hat{P}_0(y)} \tag{26}$$

where $\mathcal{V}_{\text{active}} \subseteq \mathcal{V}$ is the set of tokens observed in the production window (typically $|\mathcal{V}_{\text{active}}| \ll |\mathcal{V}|$), and $\hat{P}_t(y) = \hat{f}_t(y)/N_t$ and $\hat{P}_0(y) = \hat{f}_0(y)/N_0$ are the normalized sketch estimates.

The summation over $\mathcal{V}_{\text{active}}$ rather than the full vocabulary $\mathcal{V}$ is both computationally efficient (the sum has at most N_t nonzero terms, where N_t is the window size) and statistically motivated: tokens not observed in the production window contribute zero to $D_{KL}(P_t \| P_0)$ by the convention $0 \cdot \log(0/q) = 0$.

Table 7: Count-Min Sketch Parameters for CoRIM Reference Distribution

Parameter	Value	Rationale
$\varepsilon_{\text{sketch}}$	10^{-4}	Frequency error $\leq 0.01\%$ of total count
δ_{sketch}	10^{-3}	99.9% confidence in error bound
Width w	27,183	$= \lceil e/10^{-4} \rceil$
Depth d_{hash}	7	$= \lceil \ln(1/10^{-3}) \rceil$
Storage	≈ 760 KB	$27{,}183 \times 7 \times 4$ bytes (32-bit counters)

Calibrated parameters for Clause AI-6. A 760 KB sketch is lightweight enough to embed directly in the CoRIM artifact alongside the model's attestation evidence, enabling any verifier to independently compute drift-kl without access to the raw validation data.

5.2.2 HyperLogLog for Vocabulary Support Monitoring

HyperLogLog (Flajolet et al., 2007) is a probabilistic cardinality estimator that counts the number of distinct elements in a data stream using logarithmic space.

Definition 5.2 (HyperLogLog)**.** *A HyperLogLog sketch with precision parameter p uses $m = 2^p$ registers, each storing the maximum position of the leading zero in a hash of the observed elements. The cardinality estimate is:*

$$\hat{n} = \alpha_m \cdot m^2 \cdot \left(\sum_{j=1}^{m} 2^{-M[j]} \right)^{-1} \tag{27}$$

where $M[j]$ is the value in register j and α_m is a bias-correction constant.

Theoretical guarantees.

- Relative error: $\sigma/\hat{n} \approx 1.04/\sqrt{m}$.
- Space complexity: $O(m \cdot \log \log n)$ bits, where n is the true cardinality. With $p = 14$ ($m = 16{,}384$ registers), the sketch occupies approximately 12 KB and estimates cardinality with $\sim$0.8% relative error.

Application to drift monitoring. HyperLogLog is not used to estimate token *frequencies* (that is the Count-Min Sketch's role) but to estimate the *support* of the distribution—the number of distinct tokens active in the production stream. By comparing the HLL sketch of the production window with the HLL sketch stored in the CoRIM, the system detects vocabulary support shifts:

- **Support expansion**: The model is using tokens that were not in the active vocabulary during validation. This can indicate language switching, code generation in a text context, or the emergence of novel token patterns.
- **Support contraction**: The model's active vocabulary has shrunk, potentially indicating mode collapse or inertial drift (Section 2.2).

The HLL comparison is a fast ($O(m)$) pre-filter: if the vocabulary support has not changed, the system can skip the more expensive Count-Min Sketch divergence computation for that window. If support has changed, the full drift-kl computation is triggered.

Table 8: HyperLogLog Parameters for CoRIM Vocabulary Support Monitoring

Parameter	Value	Rationale
Precision p	14	Standard precision; 0.8% relative error
Registers m	16,384	$= 2^{14}$
Storage	$\approx$ 12 KB	16,384 $\times$ 6 bits per register

Calibrated parameters for Clause AI-6.

5.2.3 Sketch Composition for Bigram and Trigram Monitoring

For bigram monitoring, the Count-Min Sketch is applied to the space of token pairs (y_t, y_{t-1}). Rather than hashing into the intractable space $\mathcal{V}^2$, the system hashes the *concatenated token indices*:

$$h_j(y_t, y_{t-1}) = h_j(y_t \cdot |V| + y_{t-1}) \tag{28}$$

This maps the bigram space into the same sketch dimensions as the unigram space, with the Count-Min Sketch's error guarantees preserved (the hash functions are defined over arbitrary integers, not just vocabulary indices). The same approach extends to trigrams with $h_j(y_t \cdot |V|^2 + y_{t-1} \cdot |V| + y_{t-2})$.

The trade-off is increased collision probability: the effective universe size increases from $|V|$ to $|V|^2$ (bigrams) or $|V|^3$ (trigrams), which increases the expected error for a fixed sketch width. To maintain the same error bound, the sketch width must increase proportionally. In practice, the system uses separate sketch instances for each n-gram level, with widths calibrated to the effective universe size:

Table 9: Count-Min Sketch Parameters by N-gram Level

Level	$\varepsilon_{\text{sketch}}$	Width w	Depth	Storage
Unigram	10^{-4}	27,183	7	$\approx$ 760 KB
Bigram	10^{-3}	2,719	7	$\approx$ 76 KB
Trigram	10^{-2}	272	7	$\approx$ 7.6 KB

The relaxed error bounds for higher-order n-grams reflect the governance trade-off: unigram drift requires precise detection (it is the mandatory invariant); bigram and trigram drift monitoring provides supplementary diagnostic resolution and can tolerate coarser approximation.

5.3 CoRIM Artifact Format

The Concise Reference Integrity Manifest (CoRIM) is defined by the IETF draft-ietf-rats-corim specification as part of the Remote ATtestation procedureS (RATS) architecture. CoRIM provides a CBOR-encoded, COSE-signed container for reference values that a Verifier uses to appraise attestation Evidence.

The Auburn Governance Stack extends CoRIM with distribution-specific content types for the drift monitoring reference.

5.3.1 CoRIM Extension: Distribution Reference Value

Listing 1: CoRIM Distribution Reference Value (CDDL)

```
; Auburn CoRIM extension: Distribution Reference Value
auburn-dist-ref = {
    ; Reference distribution type
    dist-type: dist-type-enum,

    ; Model identity binding
    model-id: model-identity,

    ; Deployment configuration binding
    deployment-config: {
        precision: precision-enum,      ; FP32, FP16, INT8, INT4
        quantization-scheme: tstr,      ; e.g., "GPTQ", "AWQ", "none"
        hardware-platform: tstr,        ; e.g., "A100-80GB", "H100-SXM"
    },

    ; Sketch representations
    sketches: {
        ; Mandatory: unigram Count-Min Sketch
        unigram-cms: cms-sketch,

        ; Mandatory: vocabulary support HLL
        vocab-hll: hll-sketch,

        ; Optional: higher-order sketches
        ? bigram-cms: cms-sketch,
        ? trigram-cms: cms-sketch,
    },

    ; Smoothing parameters (for exact reproduction)
    smoothing: {
        method: smoothing-enum,         ; "absolute-discount", "laplace
            ", "good-turing"
        discount-param: float,          ; e.g., 0.5 for absolute
            discounting
        floor-epsilon: float,           ; minimum probability mass
    },

    ; Hidden state reference (optional)
    ? hidden-state-ref: {
        layer-index: uint,              ; which layer's activations
```

```
        dimensionality: uint,          ; hidden state dimension
        reference-bank-hash: hash-entry, ; hash of the reference bank
            file
        reference-bank-uri: tstr,      ; URI to retrieve the reference
            bank
        drift-head-hash: hash-entry,   ; hash of the trained drift head
            weights
        drift-head-uri: tstr,          ; URI to retrieve the drift head
    },

    ; Construction metadata
    construction: {
        strategy: strategy-enum,       ; "validation", "training", "
            synthetic"
        dataset-hash: hash-entry,      ; hash of the dataset used
        dataset-size: uint,            ; number of tokens in
            construction set
        construction-date: tdate,
        smoothing-applied: bool,
    },

    ; Provenance
    parent-corim: ? corim-locator,     ; link to parent (for versioned
        baselines)
    sentinel-evidence: ? sep-ref,      ; Sentinel Evidence Package
        reference
}

dist-type-enum = &(
    VALIDATION: 0,
    TRAINING: 1,
    SYNTHETIC: 2,
)

precision-enum = &(
    FP32: 0,
    FP16: 1,
    BF16: 2,
    INT8: 3,
    INT4: 4,
)

smoothing-enum = &(
    ABSOLUTE-DISCOUNT: 0,
    LAPLACE: 1,
    GOOD-TURING: 2,
)

strategy-enum = &(
    VALIDATION: 0,
    TRAINING: 1,
    SYNTHETIC: 2,
    COMPOSITE: 3,        ; combination of strategies
)

cms-sketch = {
    width: uint,
    depth: uint,
```

```
    counters: bstr,           ; serialized counter matrix
    total-count: uint,        ; total observations
    hash-seeds: [+ uint],     ; seeds for pairwise-independent hash
        functions
}

hll-sketch = {
    precision: uint,          ; p parameter
    registers: bstr,          ; serialized register array
}
```

5.3.2 Deployment Configuration Binding

A critical requirement of the CoRIM distribution reference is the `deployment-config` block. This binds the reference distribution to the *exact deployment configuration* of the attested model:

Deployment Configuration Binding

The CoRIM distribution reference **MUST** be constructed from the model operating in the identical configuration (precision, quantization scheme, hardware platform) as the production deployment. A reference constructed from a FP16 model **MUST NOT** be used to monitor a INT4-quantized deployment of the same model. Mismatched configurations produce a persistent baseline offset (Section 2.6) that conflates quantization artifacts with genuine operational drift.

This requirement has a practical consequence: if a model is deployed in multiple configurations (e.g., FP16 on A100 for batch processing and INT4 on edge devices for real-time inference), each configuration **MUST** have its own CoRIM distribution reference. The `model-id` field identifies the model; the `deployment-config` block identifies the specific configuration.

5.4 The Reilly Sentinel Protocol: Baseline Versioning

Models are not static. They are fine-tuned, quantized, updated, and retrained. Each of these events alters the model's output distribution, necessitating a new baseline. Without a formal versioning protocol, the baseline management process is vulnerable to *drift washing*: the practice of simply resetting the baseline to the current (drifted) state to silence alarms without governance review.

The Reilly Sentinel Protocol (RSP) manages the lifecycle of CoRIM distribution references, ensuring that every baseline transition is justified, documented, and cryptographically linked to its predecessor.

5.4.1 Protocol Overview

The RSP operates on three principles:

1. **Every baseline transition requires justification.** A new CoRIM distribution reference cannot be issued unless accompanied by a Sentinel Evidence Package (SEP) documenting the reason for the update, the data used to construct the new reference, and the validation metrics confirming that the updated model meets deployment requirements.

2. **Every baseline has provenance.** Each CoRIM distribution reference carries a `parent-corim` locator linking it to the previous reference. This creates an immutable, append-only chain of baseline versions, analogous to a Git commit history for the model's behavioral envelope.

3. **Baseline transitions are auditable events.** The act of updating the baseline is itself a governance event that **MUST** be recorded in the model's Decision Receipt chain (Layer 3 provenance). An auditor can reconstruct the complete history of baseline transitions and evaluate whether each transition was justified.

5.4.2 Sentinel Evidence Package (SEP)

Definition 5.3 (Sentinel Evidence Package)**.** *A Sentinel Evidence Package (SEP) is a signed artifact accompanying a CoRIM baseline update that contains the following mandatory fields:*

1. ***Update reason**: Enumerated type—`FINE_TUNING`, `QUANTIZATION`, `RETRAINING`, `CONFIGURATION_CHANGE`, `SCHEDULED_REFRESH`.*
2. ***Training/update logs**: Hash of the training logs, fine-tuning configuration, learning rate schedule, and number of update steps.*
3. ***Validation metrics**: The model's performance on the standard validation suite after the update, including per-benchmark scores and the drift-kl value between the old and new reference distributions.*
4. ***Cross-reference drift**: $D_{KL}(P_0^{new} \| P_0^{old})$—the KL divergence between the new and old reference distributions. This quantifies how much the baseline has shifted and provides an auditable record of cumulative baseline drift over the model's lifecycle.*
5. ***Approval record**: Identifier of the governance authority (human or automated) that approved the baseline transition.*

5.4.3 Anti-Drift-Washing Mechanism

The RSP prevents drift washing through three mechanisms:

Mechanism 1: Cross-reference drift accumulation. Each SEP records $D_{KL}(P_0^{\text{new}} \| P_0^{\text{old}})$. The cumulative drift across all baseline transitions is computable from the SEP chain:

$$D_{\text{cumulative}} = \sum_{i=1}^{n} D_{KL}(P_0^{(i)} \| P_0^{(i-1)}) \tag{29}$$

If the cumulative baseline drift exceeds a governance threshold $D_{\text{lifecycle}}$, the system flags the model as having drifted beyond its original validation envelope—even though each individual baseline transition was within bounds. This detects the "boiling frog" attack: a series of small, individually justified baseline updates that cumulatively transform the model's behavioral profile.

Mechanism 2: Immutable provenance chain. The `parent-corim` linkage is cryptographically signed. An auditor can verify that the chain is unbroken—no intermediate baselines have been deleted or replaced. Any gap in the chain indicates tampering.

Mechanism 3: Validation metric regression detection. The SEP includes the model's validation metrics after each update. If validation performance has degraded beyond a tolerance threshold, the SEP is flagged as requiring governance escalation—the baseline update may be silencing drift alarms at the cost of reduced model quality.

5.4.4 Lifecycle Management

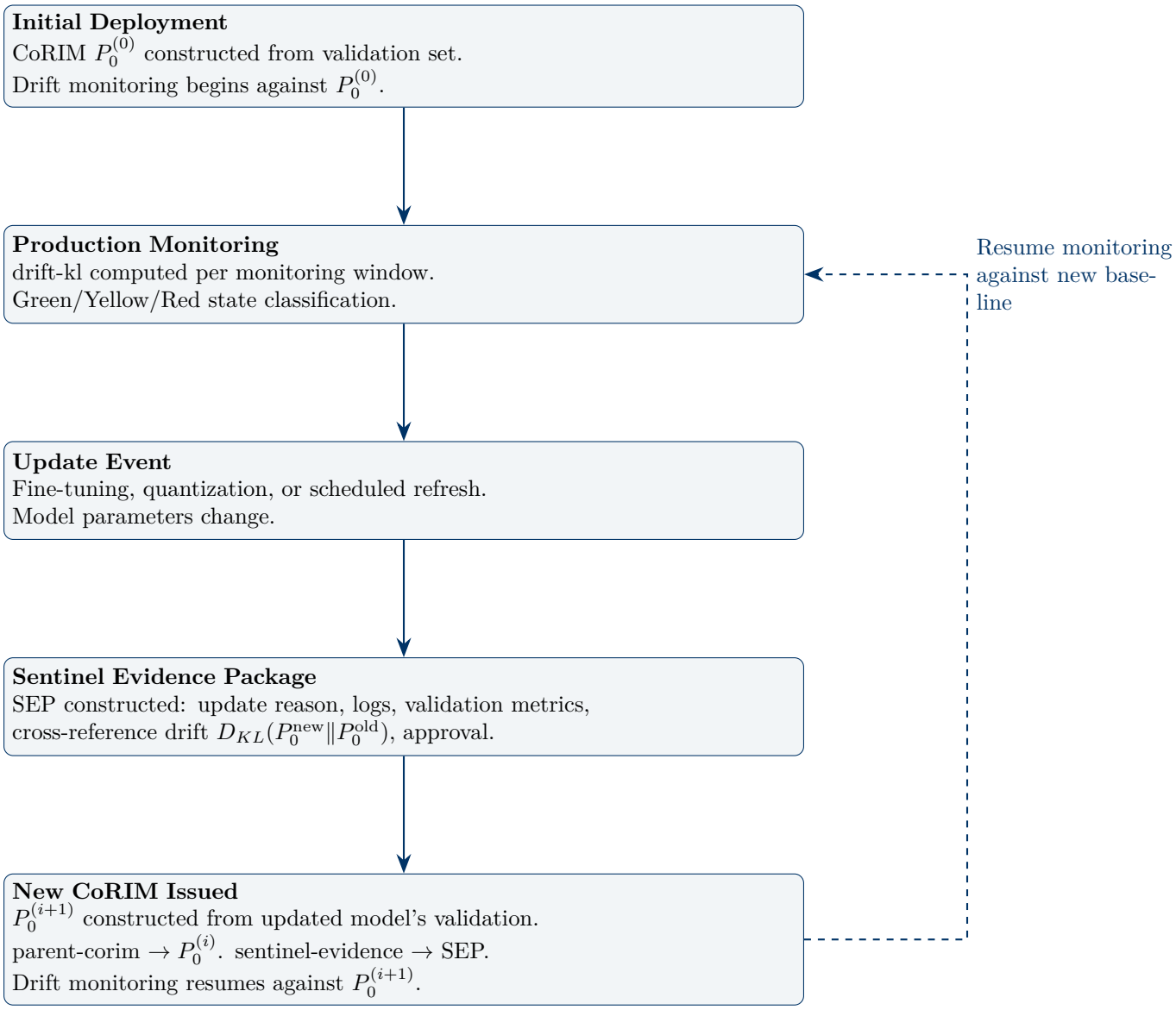

Figure 1: Reilly Sentinel Protocol: Baseline Versioning Lifecycle

5.5 CoRIM Reference Construction: Complete Procedure

The following procedure specifies the end-to-end construction of a CoRIM distribution reference for a new model deployment.

Step 1: Configuration lock. The model's deployment configuration (precision, quantization scheme, hardware platform) **MUST** be finalized before reference construction begins. Any subsequent configuration change invalidates the reference.

Step 2: Validation inference. The model is run in inference mode over the validation set $\mathcal{D}_{\text{val}}$. For each prompt $x_i \in \mathcal{D}_{\text{val}}$, the model generates a response y_i. The complete set of output tokens $\{y_i\}$ constitutes the reference corpus.

Step 3: Frequency computation. Unigram, bigram, and trigram token frequencies are computed over the reference corpus. Simultaneously, hidden state activations are collected from the designated monitoring layer (default: penultimate transformer block) to populate the hidden state reference bank.

Step 4: Smoothing. Absolute discounting with $d = 0.5$ is applied to the unigram frequency distribution. The smoothing floor $\varepsilon = d/(N_{\text{val}} + d \cdot |\mathcal{V}|)$ ensures $P_0(y) > 0$ for all $y \in \mathcal{V}$.

Step 5: Sketch construction. Count-Min Sketches are constructed for unigram, bigram, and trigram levels using the parameters in Table 9. HyperLogLog sketch is constructed from the active token set.

Step 6: Drift head training (optional). If hidden state monitoring is enabled, the Donsker-Varadhan drift head (Section 4.3.2) is trained using the reference bank and held-out validation states.

Step 7: Serialization. All sketch data, smoothing parameters, hidden state reference metadata, deployment configuration, and construction metadata are serialized into the CoRIM artifact using the CDDL schema in Listing 1.

Step 8: Signing. The CoRIM artifact is signed using COSE_Sign1 with the model vendor's attestation key. The signature binds the reference distribution to the model identity and deployment configuration, preventing substitution or tampering.

Step 9: Registration. The signed CoRIM is registered in the SCITT (Supply Chain Integrity, Transparency, and Trust) append-only ledger, creating an immutable record of the reference distribution's existence and content.

CoRIM Construction Completeness

A CoRIM distribution reference is considered complete—and the model is eligible for MAI-1 conformant attestation—only when all nine steps have been executed and the signed artifact has been registered in the SCITT ledger. Deployments lacking a registered CoRIM distribution reference **MUST NOT** claim Clause AI-6 conformance.

6 Mathematical Formalization of Clause AI-6

The preceding sections established the empirical necessity (Section 2), the taxonomic framework (Section 3), the information-theoretic foundations (Section 4), and the baseline construction methodology (Section 5) for distribution drift monitoring. This section presents the formal mathematical statement of Clause AI-6: the invariant definition, the threshold calibration methodology linking drift magnitude to governance risk tolerance, the compliance state transition logic, and the complete audit-ready parameter table.

This section is the normative core of the document. Every requirement stated here is binding on MAI-1-conformant deployments and testable by the CTS-1 conformance test suite (assertions AS-SM-L2-10 through TE-SM-L2-10.04).

6.1 The Distribution Drift Invariant

Clause 1 (Clause AI-6: Distribution Drift Bound)**.** *Let $\mathcal{M}$ be a foundation model deployed in production with validated baseline distribution P_0 sealed in a CoRIM artifact (Section 5). Let P_t*

denote the empirical output distribution of $\mathcal{M}$ over a sliding monitoring window centered at time t. The distribution drift invariant requires:

$$\boxed{D_{KL}(P_t \| P_0) \leq D_{\max} \quad \forall\, t \in [t_0, t_f]} \tag{30}$$

where:

- *$D_{KL}(P_t \| P_0)$ is the Kullback–Leibler divergence from the baseline distribution P_0 to the production distribution P_t, computed using the estimation methods specified in Section 4;*
- *$D_{\max} > 0$ is the certified maximum permissible divergence, calibrated per Section 6.3 and carried in the MAI-1 Layer 2 field `thresholds.drift-kl-max`;*
- *$[t_0, t_f]$ is the attestation validity period;*
- *the monitoring window has width W tokens (calibrated per deployment context; see Section 6.2).*

Remark 6.1. *Equation 30 is the restatement of MAI-1 Definition 7.3. The contribution of this document is not the invariant statement itself—which is normatively defined in MAI-1—but the complete derivation of $D_{\max}$, the specification of the monitoring window W, the calibration methodology, and the compliance state transition logic that operationalizes the invariant.*

6.2 Monitoring Window Specification

The monitoring window W determines the granularity at which drift is measured. A window that is too narrow produces noisy estimates dominated by sampling variance; a window that is too wide smooths out abrupt drift events that require immediate detection. The calibration must balance statistical reliability against detection latency.

6.2.1 Statistical Reliability Constraint

The empirical distribution P_t is estimated from a finite sample of W tokens. The sampling variance of the KL divergence estimator places a lower bound on the window size.

For a categorical distribution over $|V|$ states estimated from W i.i.d. samples, the variance of the plugin KL divergence estimator is approximately:

$$\mathrm{Var}[\hat{D}_{KL}] \approx \frac{1}{W} \sum_{y \in \mathcal{V}} P_0(y) \left(\log \frac{P_0(y)}{P_t(y)} \right)^2 - \frac{1}{W} \left(D_{KL}(P_t \| P_0) \right)^2 \tag{31}$$

Under the null hypothesis of no drift ($P_t = P_0$), this simplifies to:

$$\mathrm{Var}[\hat{D}_{KL}] \approx \frac{1}{W} \cdot \mathrm{Var}_{y \sim P_0} \left[\log \frac{P_0(y)}{P_0(y)} \right] = 0 \tag{32}$$

which is degenerate. The practical variance under near-null conditions (small drift) is dominated by the finite-sample bias of the plugin estimator:

$$\mathrm{Bias}[\hat{D}_{KL}] \approx \frac{|\mathcal{V}_{\text{eff}}| - 1}{2W} \tag{33}$$

where $|\mathcal{V}_{\text{eff}}|$ is the effective vocabulary size (the number of tokens with non-negligible probability under P_0). This bias term sets the noise floor of the estimator: for drift-kl measurements to be meaningful, the true drift must exceed the bias.

Minimum Window Size

The monitoring window W **MUST** satisfy:

$$W \geq \frac{|\mathcal{V}_{\text{eff}}| - 1}{2 \cdot D_{\text{resolution}}} \tag{34}$$

where $D_{\text{resolution}}$ is the minimum drift magnitude that the system must reliably detect. For governance purposes, $D_{\text{resolution}} = 0.01$ nats (one-tenth of the Yellow threshold).

For a model with $|\mathcal{V}_{\text{eff}}| \approx 10{,}000$ active tokens (a typical effective vocabulary for a domain-specific deployment), this yields $W \geq (10{,}000 - 1)/(2 \times 0.01) = 499{,}950$ tokens. This is a large window—approximately 375,000 words or several hundred pages of text.

For general-purpose models with $|\mathcal{V}_{\text{eff}}| \approx 50{,}000$, the requirement increases to $W \geq 2{,}499{,}950$ tokens, which may span hours or days of production traffic.

6.2.2 Detection Latency Constraint

The window size W also determines the *detection latency*: the number of tokens that must be generated after a drift event before the event is detectable in the windowed estimate. For an abrupt drift event occurring at time t^*, the windowed KL divergence is a weighted average of pre-drift (in-distribution) and post-drift (out-of-distribution) tokens:

$$\hat{D}_{KL}(t) \approx \frac{t - t^*}{W} \cdot D_{\text{true}} \quad \text{for } t \in [t^*, t^* + W] \tag{35}$$

The drift exceeds the threshold $D_{\max}$ when $\frac{t-t^*}{W} \cdot D_{\text{true}} > D_{\max}$, i.e., after approximately $\frac{D_{\max}}{D_{\text{true}}} \cdot W$ tokens. For a severe drift event ($D_{\text{true}} = 1.0$ nats) with $D_{\max} = 0.3$ nats and $W = 500{,}000$ tokens, the detection delay is approximately 150,000 tokens—unacceptable for safety-critical applications.

6.2.3 Resolution: Multi-Scale Windowing

The tension between statistical reliability (large W) and detection latency (small W) is resolved by *multi-scale windowing*: the Drift Monitor maintains multiple concurrent windows at different scales, each calibrated for a different class of drift event.

Table 10: Multi-Scale Monitoring Window Configuration

Window	Size W	Target Drift Type	Detection Latency	False
W_{fast}	1,000 tokens	Abrupt concept drift, adversarial injection	< 300 tokens	Higher
W_{medium}	50,000 tokens	Inertial drift, composite burst	$\sim 15{,}000$ tokens	Modera
W_{slow}	500,000 tokens	Gradual prior probability shift, RLHF artifacts	$\sim 150{,}000$ tokens	Low (h

The compliance determination uses the *most conservative* (largest drift) value across all windows:

$$D_{KL}^{\text{reported}}(t) = \max\left\{ D_{KL}^{W_{\text{fast}}}(t),\ D_{KL}^{W_{\text{medium}}}(t),\ D_{KL}^{W_{\text{slow}}}(t) \right\} \tag{36}$$

This ensures that abrupt events are detected quickly (via W_{fast}) while gradual shifts are detected reliably (via W_{slow}). The value reported in the `drift-kl` field of the MAI-1 Layer 2 payload is $D_{KL}^{\text{reported}}(t)$, the maximum across all active windows.

Remark 6.2. *The fast window $W_{fast} = 1{,}000$ tokens has high sampling variance and will produce false positive exceedances of $D_{\max}$ under normal conditions. This is acceptable because W_{fast}*

is processed by the Page-Hinkley test (Section 7.3), which is specifically designed for sequential change detection with controlled false alarm rates. A single exceedance in W_{fast} does not trigger a compliance state transition; it triggers escalated monitoring (Section 6.4).

6.3 Threshold Calibration Methodology

The certified threshold $D_{\max}$ is the single most governance-critical parameter in Clause AI-6. A threshold that is too tight produces excessive false positives, rendering the monitoring system operationally useless. A threshold that is too loose permits genuine drift to accumulate undetected, defeating the purpose of the invariant.

The calibration methodology derives $D_{\max}$ from the *natural variation* of the model's output distribution under normal operating conditions, augmented by a governance risk margin.

6.3.1 Step 1: Empirical Null Distribution

During CoRIM construction (Section 5.5, Step 2), the validation inference produces a stream of output tokens. This stream is partitioned into non-overlapping windows of size W_{slow}. For each window i, the KL divergence against the full-validation reference is computed:

$$D_i = D_{KL}(\hat{P}_i \| P_0), \quad i = 1, \ldots, N_{\text{windows}} \tag{37}$$

The set $\{D_1, D_2, \ldots, D_{N_{\text{windows}}}\}$ constitutes the *empirical null distribution* of drift-kl: the distribution of drift values that a healthy, in-distribution model produces due to natural sampling variation alone.

Expected properties of the null distribution.

- **Mean**: $\bar{D} \approx \frac{|\mathcal{V}_{\text{eff}}|-1}{2W}$ (the finite-sample bias; Equation 33). For $|\mathcal{V}_{\text{eff}}| = 10{,}000$ and $W = 500{,}000$: $\bar{D} \approx 0.01$ nats.
- **Standard deviation**: σ_D depends on the entropy of the baseline distribution and the window size. Empirically, $\sigma_D \approx 0.005$–0.02 nats for typical LLM deployments.
- **Shape**: Approximately chi-squared with $|\mathcal{V}_{\text{eff}}| - 1$ degrees of freedom, scaled by $1/(2W)$. For large $|\mathcal{V}_{\text{eff}}|$, the Central Limit Theorem applies and the distribution is approximately Gaussian.

6.3.2 Step 2: Conformal Calibration

Rather than assuming a parametric form for the null distribution, the calibration uses *conformal prediction* (Section 8) to provide a distribution-free threshold. The conformal approach guarantees coverage without assumptions on the shape of the null distribution.

The procedure:

1. Compute the null drift values $\{D_1, \ldots, D_{N_{\text{windows}}}\}$ from Step 1.
2. Sort them in ascending order: $D_{(1)} \leq D_{(2)} \leq \cdots \leq D_{(N_{\text{windows}})}$.
3. For a desired false positive rate α_{FP}, set the threshold as the $(1 - \alpha_{\text{FP}})$-quantile:
$$D_{\text{conformal}} = D_{(\lceil (1-\alpha_{\text{FP}})(N_{\text{windows}}+1) \rceil)} \tag{38}$$
4. By the conformal guarantee, this threshold satisfies:
$$\Pr[D_{KL}(P_t \| P_0) > D_{\text{conformal}} \mid P_t = P_0] \leq \alpha_{\text{FP}} \tag{39}$$
for any new observation $D_{KL}(P_t \| P_0)$ drawn from the same (null) distribution.

This is the *distribution-free* threshold: it provides the false positive rate guarantee without assuming that the null distribution is Gaussian, chi-squared, or any other parametric family.

6.3.3 Step 3: Governance Risk Margin

The conformal threshold $D_{\text{conformal}}$ controls the false positive rate under null conditions. The governance risk margin Δ_{gov} provides additional headroom for input distribution variation that is not captured by the validation set:

$$D_{\max} = D_{\text{conformal}} + \Delta_{\text{gov}} \tag{40}$$

The governance risk margin is calibrated by deployment context:

Table 11: Governance Risk Margin by Deployment Context

Deployment Context	$\alpha_{\mathbf{FP}}$	$\Delta_{\mathbf{gov}}$ **(nats)**	**Rationale**
Defense / safety-critical	0.001	0.02	Tight margin; false negatives are unacceptable
FDA SaMD / medical	0.005	0.03	Moderate margin; PCCP compliance requires sensitivity
Financial services (SR 11-7)	0.01	0.05	Standard margin; aligned with PSI thresholds
General enterprise	0.01	0.10	Relaxed margin; operational flexibility
Research / non-regulated	0.05	0.20	Wide margin; monitoring is advisory, not binding

6.3.4 Concrete Threshold Calibration Examples

Table 12: Concrete $D_{\max}$ Calibration for Representative Deployments

Deployment	$\lvert\mathcal{V}_{\mathbf{eff}}\rvert$	$\bar{D}$ **(nats)**	$D_{\mathbf{conformal}}$	$\Delta_{\mathbf{gov}}$	$D_{\max}$ **(nats)**
Medical chatbot (7B, INT8)	8,000	0.008	0.025	0.03	0.055
Financial analyst (70B, FP16)	15,000	0.015	0.042	0.05	0.092
General assistant (70B, FP16)	30,000	0.030	0.071	0.10	0.171
Code generation (13B, INT4)	12,000	0.012	0.035	0.10	0.135
Research prototype	50,000	0.050	0.110	0.20	0.310

Remark 6.3. *The concrete values in Table 12 are illustrative. The actual $D_{conformal}$ value depends on the specific model's validation distribution and the empirical null distribution computed during CoRIM construction. The procedure is deterministic: given the validation data, the threshold is uniquely determined by the choice of α_{FP} and Δ_{gov}.*

6.4 Compliance State Transition Logic

The drift invariant produces a continuous-valued signal ($D_{KL}(P_t \| P_0)$). Governance requires a discrete compliance determination. The compliance state machine maps the continuous drift signal to the three-state classification defined in MAI-1:

Definition 6.1 (Compliance States for Clause AI-6)**.**

- ***GREEN*** *(compliance-status = 0): $D_{KL}^{reported}(t) \leq D_{yellow}$. All monitoring windows report drift within normal bounds. Standard operation continues.*
- ***YELLOW*** *(compliance-status = 1): $D_{yellow} < D_{KL}^{reported}(t) \leq D_{\max}$. At least one monitoring window reports drift exceeding the warning threshold but below the hard invariant bound. Monitoring frequency is escalated. The `breached-invariant` field is not set (the invariant is not breached; only the warning threshold is crossed).*
- ***RED*** *(compliance-status = 2): $D_{KL}^{reported}(t) > D_{\max}$, or sustained Yellow exceeding the critical duration Δt_{crit}. The distribution drift invariant is breached. The `breached-invariant` field reports `"drift-kl"`. Automatic intervention is triggered.*

6.4.1 Warning Threshold

The warning threshold D_{yellow} is set as a fraction of $D_{\max}$:

$$D_{\text{yellow}} = \beta \cdot D_{\max}, \quad \beta \in [0.5, 0.8] \tag{41}$$

The default value is $\beta = 0.6$. This provides an early warning buffer: the system enters Yellow when drift has consumed 60% of the available headroom, providing time for investigation and potential intervention before the hard bound is reached.

6.4.2 State Transition Rules

The state transitions are governed by the following rules, evaluated at each monitoring epoch (the frequency at which drift-kl is reported in the MAI-1 artifact):

T1: GREEN → YELLOW: Triggered when $D_{KL}^{\text{reported}}(t) > D_{\text{yellow}}$ at any monitoring epoch. Transition is immediate (single exceedance triggers).

T2: YELLOW → GREEN: Triggered when $D_{KL}^{\text{reported}}(t) \leq D_{\text{yellow}}$ for N_{recovery} consecutive monitoring epochs. The recovery requirement ($N_{\text{recovery}} = 5$ by default) prevents oscillation between states due to noise.

T3: YELLOW → RED (threshold breach): Triggered when $D_{KL}^{\text{reported}}(t) > D_{\max}$ at any monitoring epoch. Transition is immediate.

T4: YELLOW → RED (sustained warning): Triggered when the system has been in YELLOW state continuously for Δt_{crit} monitoring epochs without returning to GREEN. This detects "slow burn" drift that remains just below $D_{\max}$ but persists indefinitely.

T5: RED → YELLOW: Triggered when $D_{KL}^{\text{reported}}(t) \leq D_{\max}$ for N_{recovery} consecutive monitoring epochs *and* the root cause analysis has been documented in the Decision Receipt chain.

T6: RED → GREEN: Not permitted as a direct transition. The system must pass through YELLOW and satisfy the T2 recovery condition. This prevents premature return to normal operation after a severe drift event.

6.4.3 Sustained Warning Duration

The critical duration Δt_{crit} for the T4 transition (sustained YELLOW → RED) is calibrated by deployment context:

Table 13: Sustained Warning Critical Duration by Deployment Context

Deployment Context	Δt_{crit} (epochs)	Approximate Real Time
Defense / safety-critical	10	∼10 minutes at 1 epoch/min
FDA SaMD / medical	30	∼30 minutes
Financial services	60	∼1 hour
General enterprise	180	∼3 hours
Research / non-regulated	720	∼12 hours

6.4.4 State Machine Diagram

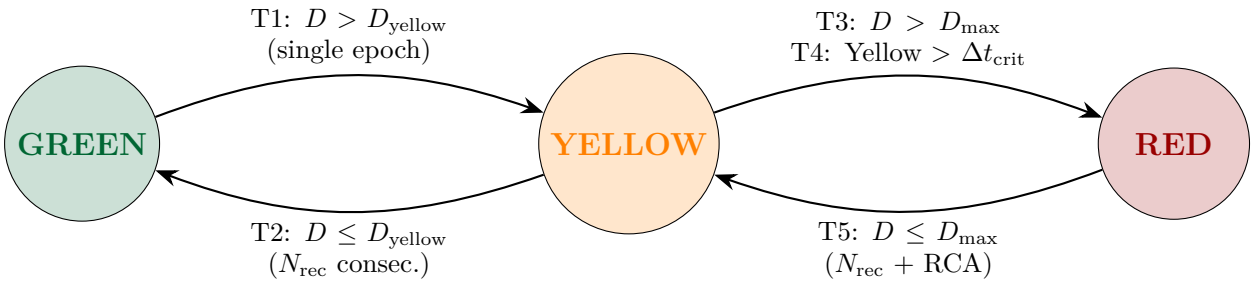

Figure 2: Clause AI-6 Compliance State Machine. T1–T5 are the transition rules defined in Section 6.4. RCA = Root Cause Analysis documented in Decision Receipt chain.

6.5 The Drift Entropy Extension

In addition to the mandatory drift-kl invariant (Equation 30), Clause AI-6 defines the drift entropy metric (Definition 3.7) as a recommended extension for high-risk deployments.

6.5.1 Formal Invariant

$$\delta_t = \sum_{k=1}^{T} D_{KL}\left(P_k^{(0)} \| P_k^{(t)}\right) \leq \delta_{\max} \quad \forall\, t \in [t_0, t_f] \tag{42}$$

where $\delta_{\max}$ is the certified maximum permissible drift entropy.

6.5.2 Threshold Calibration

The drift entropy threshold is calibrated from the hallucination correlation result: $\rho(\delta_t, \text{hallucination severity}) = 0.87$. The calibration procedure:

1. During validation, the model generates N_{gen} multi-step responses to a diverse prompt set.
2. For each response, the drift entropy δ_t is computed at each generation step t.
3. The responses are independently rated for hallucination severity on a 0–5 scale by human evaluators or a calibrated automated detector.
4. The relationship $\delta_t = f(\text{hallucination severity})$ is characterized empirically, typically exhibiting a sigmoid transition: low drift entropy ($\delta_t < 0.5$) corresponds to hallucination severity ≤ 1 (minor); high drift entropy ($\delta_t > 2.0$) corresponds to severity ≥ 4 (severe).
5. The threshold $\delta_{\max}$ is set at the point where the expected hallucination severity exceeds the acceptable level for the deployment context.

Table 14: Drift Entropy Threshold by Deployment Risk Level

Risk Level	$\delta_{\max}$ **(nats)**	**Expected Hallucination Severity**
Safety-critical (defense, medical)	0.5	≤ 1 (minor factual imprecision)
Regulated (financial, legal)	1.0	≤ 2 (moderate inconsistency)
Enterprise (general business)	2.0	≤ 3 (noticeable but non-critical)
Research / non-regulated	5.0	Advisory only

6.5.3 Integration with drift-kl

Drift entropy and drift-kl are complementary metrics. Drift-kl measures the distance of the output distribution from the *external* baseline P_0. Drift entropy measures the distance of the model's current generation plan from its *own initial plan* $P^{(0)}$. The two metrics can diverge:

- **High drift-kl, low** δ_t: The model is consistently producing output that differs from baseline but is internally coherent. Indicates prior probability shift (e.g., RLHF artifacts) or deliberate behavioral change. Governance concern: moderate (the model is consistent but different).
- **Low drift-kl, high** δ_t: The model's aggregate output distribution is close to baseline, but individual responses are internally inconsistent. Indicates planning drift and hallucination within individual episodes. Governance concern: high (the model appears healthy in aggregate but is unreliable per-response).
- **High drift-kl, high** δ_t: Both external and internal coherence are degraded. Indicates severe concept drift or approaching Evans' Law coherence limit. Governance concern: critical.

The MAI-1 Layer 2 payload carries drift entropy in the optional extension field `invariants.drift-entropy`. When present, the CTS-1 conformance test suite applies analogous assertions to those for drift-kl.

6.6 Relationship to CTS-1 Conformance Assertions

The formalization in this section directly supports the following CTS-1 assertions:

Table 15: Clause AI-6 Formalization to CTS-1 Assertion Mapping

CTS-1 Assertion	AI-6 Section	Relationship
TE-SM-L2-10.01	§6.1, §6.3	Verifies `drift-kl` and `drift-kl-max` are present, correctly typed, and $D_{\max} > 0$.
TE-SM-L2-10.02	§6.1	Verifies $D_{KL} \geq 0$ (Gibbs' Inequality, Theorem 4.1). Negative values indicate measurement error.
TE-SM-L2-10.03	§6.4	Verifies compliance flag consistency: if $D_{KL} > D_{\max}$, then compliance-status **MUST** be YELLOW or RED.
TE-SM-L2-10.04	§6.3	TSPRT with $H_0 : \mu_D \leq D_{\max}$. The threshold $D_{\max}$ is calibrated per §6.3; the TSPRT parameters ($\alpha \leq 0.01$, $\beta \leq 0.05$, $N_{\max} = 100$) are validated against the empirical null distribution of §6.3.1.
VE-SM-L2-10.01	§5, §6.2	Vendor documents drift measurement methodology: baseline construction strategy, CoRIM reference, measurement window, smoothing parameters.
VE-SM-L2-10.02	§6.3	Vendor documents $D_{\max}$ derivation: empirical null distribution, conformal calibration, governance risk margin, relationship to validated operating envelope.

6.7 Complete Audit-Ready Parameter Table

Table 16 consolidates all calibrated parameters for Clause AI-6 implementation.

Table 16: Clause AI-6: Complete Audit-Ready Parameter Table

Parameter	Symbol	Default Value	Description	Section
Invariant Parameters				
Maximum permissible drift	$D_{\max}$	Context-dependent	Certified upper bound on $D_{KL}(P_t \Vert P_0)$. Calibrated per §6.3.	§6.1
Warning threshold	D_{yellow}	$0.6 \cdot D_{\max}$	Threshold for GREEN → YELLOW transition.	§6.4
Warning fraction	β	0.6	$D_{\text{yellow}} = \beta \cdot D_{\max}$; configurable $\in [0.5, 0.8]$.	§6.4
False positive rate	α_{FP}	Context-dependent	Probability of drift-kl exceeding $D_{\max}$ under null.	§6.3.2
Governance risk margin	Δ_{gov}	Context-dependent	Additional headroom beyond conformal threshold.	§6.3
Maximum drift entropy	$\delta_{\max}$	Context-dependent	Certified upper bound on planning drift (extension).	§6.5
Monitoring Window Parameters				
Fast window	W_{fast}	1,000 tokens	Abrupt drift detection.	§6.2
Medium window	W_{medium}	50,000 tokens	Inertial drift and composite burst detection.	§6.2
Slow window	W_{slow}	500,000 tokens	Gradual prior probability shift detection.	§6.2
Drift resolution	$D_{\text{resolution}}$	0.01 nats	Minimum detectable drift magnitude.	§6.2
State Machine Parameters				
Recovery count	N_{recovery}	5 epochs	Consecutive in-bound epochs for state recovery.	§6.4
Sustained warning duration	Δt_{crit}	Context-dependent	Maximum YELLOW duration before RED escalation.	§6.4
Baseline Construction Parameters				
Smoothing method	—	Absolute discounting	Applied during CoRIM reference construction.	§4.2.1
Smoothing discount	d	0.5	Absolute discount parameter.	§4.2.1
Smoothing floor	ε	$d/(N + d \cdot \lvert V \rvert)$	Minimum probability per token in reference.	§4.2.1
CMS width (unigram)	w	27,183	Count-Min Sketch width for unigram reference.	§5.2.1
CMS depth	d_{hash}	7	Count-Min Sketch depth (all levels).	§5.2.1
CMS error bound	$\varepsilon_{\text{sketch}}$	10^{-4} (unigram)	Frequency estimation error bound.	§5.2.1
HLL precision	p	14	HyperLogLog precision parameter.	§5.2.2
Lifecycle Parameters				
Cumulative baseline drift limit	$D_{\text{lifecycle}}$	$3 \times D_{\max}$	Maximum cumulative drift across all baseline updates.	§5.4

7 Online Detection Algorithms

The drift invariant (Equation 30) defines the compliance condition. The multi-scale windowing architecture (Section 6.2) defines the measurement granularity. But the raw drift-kl signal—a noisy, streaming scalar—must be processed by algorithms that can distinguish genuine distributional shift from sampling noise, detect both gradual and abrupt changes, and provide formal bounds on false positive and false negative rates.

This section specifies the online detection algorithm suite for Clause AI-6: three complementary algorithms, each selected for a specific class of drift event, composed into a unified detection pipeline with formally characterized operating properties. The algorithms must operate on streaming data, update their internal state with bounded per-observation complexity, and be computationally feasible at the token generation rates of production LLM systems (thousands to tens of thousands of tokens per second).

7.1 Algorithm Selection Criteria

The selection of online detection algorithms for Clause AI-6 was governed by five criteria:

1. **Formal false alarm bounds.** The algorithm must provide theoretical guarantees on the false positive rate (Type I error) under the null hypothesis of no drift. Ad hoc heuristics without formal guarantees are excluded.

2. **Sensitivity to the target drift type.** Each algorithm must be optimally suited to a specific drift signature (gradual, abrupt, or distributional shape change). No single algorithm is optimal for all drift types.

3. **Bounded per-observation complexity.** The algorithm must process each new drift-kl observation in $O(1)$ amortized time (or $O(\log W)$ at worst), independent of the total number of observations processed.

4. **Adaptivity.** The algorithm should adapt its sensitivity to the local properties of the signal—tightening during stable periods and relaxing during periods of legitimate high variance.

5. **Established provenance.** The algorithm must have been published in peer-reviewed proceedings, with theoretical analysis and empirical validation on streaming data. Novel, untested algorithms are excluded from a governance specification.

Three algorithms satisfy all five criteria: ADWIN (Adaptive Windowing), the Page-Hinkley test, and KSWIN (Kolmogorov-Smirnov Windowing). Table 17 summarizes the selection.

Table 17: Online Detection Algorithm Selection for Clause AI-6

Algorithm	Target	Complexity	Provenance	Role in AI-6
ADWIN	Gradual drift	$O(\log W)$ amortized	Bifet & Gavaldà, 2007	Primary detector; always-on
Page-Hinkley	Abrupt drift	$O(1)$	Page, 1954; Hinkley, 1971	Emergency brake; catastrophic events
KSWIN	Shape change	$O(W \log W)$	Rabanser et al., 2019	Confirmation; high-rigor second opinion

7.2 ADWIN (Adaptive Windowing)

ADWIN (Bifet & Gavaldà, 2007) is the primary drift detection algorithm in the Clause AI-6 monitoring suite. It is a parameter-free, adaptive sliding window algorithm that automatically adjusts its window size to the rate of distributional change in the signal.

7.2.1 Algorithm Description

ADWIN maintains a variable-length window W of recent drift-kl observations. At each new observation, ADWIN tests whether any partition of the window into two contiguous sub-windows (W_0, W_1) exhibits a statistically significant difference in their means.

Definition 7.1 (ADWIN Change Detection)**.** *Let $\hat{\mu}_0$ and $\hat{\mu}_1$ be the sample means of sub-windows W_0 (older) and W_1 (newer) of sizes n_0 and n_1 respectively. ADWIN signals a change when:*

$$|\hat{\mu}_0 - \hat{\mu}_1| \geq \epsilon_{cut} \tag{43}$$

where the adaptive threshold is:

$$\epsilon_{cut} = \sqrt{\frac{1}{2m} \cdot \ln \frac{4n}{\delta_{ADWIN}}} \tag{44}$$

with $m = \frac{1}{1/n_0 + 1/n_1}$ (the harmonic mean of the sub-window sizes), $n = n_0 + n_1$ (the total window size), and δ_{ADWIN} the confidence parameter controlling the false alarm rate.

When a statistically significant difference is detected, ADWIN *drops the older sub-window* W_0 and continues monitoring with only the newer data. This is the adaptive mechanism: the window shrinks when drift is detected (capturing the new regime quickly) and grows during stable periods (accumulating statistical power for reliable detection).

7.2.2 Theoretical Guarantees

Theorem 7.1 (ADWIN False Alarm Bound (Bifet & Gavaldà, 2007))**.** *Under the null hypothesis of a stationary process (no drift), the probability that ADWIN signals a false alarm at any given step is at most δ_{ADWIN}.*

Theorem 7.2 (ADWIN Detection Guarantee)**.** *If the true mean of the drift-kl signal changes by at least $\epsilon > 0$ at some point in time, ADWIN will detect the change within $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta_{ADWIN}}\right)$ observations with probability $\geq 1 - \delta_{ADWIN}$.*

The detection delay scales as $1/\epsilon^2$: large drift events are detected quickly; small, gradual shifts require more observations. This is the correct behavior for governance: severe drift triggers rapid response while minor fluctuations are tolerated.

7.2.3 Computational Properties

ADWIN stores the window data in an exponential histogram (a compressed representation using $O(\log W)$ "buckets"), which provides:

- **Space complexity**: $O\left(\frac{1}{\epsilon^2} \log W\right)$ memory, where W is the current window size and ϵ is the desired precision of the mean estimate.
- **Per-observation time**: $O(\log W)$ amortized. Each new observation is inserted into the histogram and triggers at most $O(\log W)$ cut-point tests.
- **Maximum window size**: Unbounded. During stable periods, the window grows to capture the full stable history, maximizing statistical power. The exponential histogram ensures that memory grows only logarithmically with window size.

7.2.4 Why ADWIN for LLMs

ADWIN is the ideal primary detector for LLM drift monitoring because of three properties:

1. **Parameter-free operation.** ADWIN auto-tunes its window size. There is no hyperparameter specifying "how many observations to consider"—the algorithm discovers the appropriate window from the data. This eliminates the calibration burden that would otherwise require per-deployment tuning.

2. **Dual-nature handling.** ADWIN keeps a long window during gradual drift (accumulating evidence of a slow trend) but shrinks the window instantly when abrupt drift is detected (capturing the new regime). This handles the dual nature of LLM drift documented in Section 2: slow inertial drift (Section 2.2) and sudden adversarial perturbation (Section 2.7).

3. **Negligible latency.** At $O(\log W)$ per observation, ADWIN adds microseconds of processing per token—orders of magnitude below the 1–2% latency budget.

7.2.5 Calibrated Parameters

Table 18: ADWIN Parameters for Clause AI-6

Parameter	Value	Rationale
Confidence δ_{ADWIN}	0.002	Controls false alarm rate; calibrated to produce $<$ 1 false alarm per 500 monito
Clock parameter	32	Number of observations between cut-point checks; balances responsiveness and
Min window size	100	Minimum observations before testing begins; prevents noise-driven alarms duri

7.3 Page-Hinkley Test

The Page-Hinkley (PH) test is a sequential analysis method for detecting abrupt changes in the mean of a signal. It is the "emergency brake" in the Clause AI-6 detection pipeline, designed to catch catastrophic drift events that require immediate intervention.

7.3.1 Algorithm Description

Definition 7.2 (Page-Hinkley Test)**.** *The Page-Hinkley test maintains two cumulative statistics:*

$$m_T = \sum_{t=1}^{T} (x_t - \bar{x}_T - \nu) \tag{45}$$

$$M_T = \min_{1 \le t \le T} m_t \tag{46}$$

where x_t is the drift-kl observation at step t, $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ is the running mean, and $\nu \ge 0$ is the allowance parameter *(also called the detection threshold or magnitude of allowed change).*

The test signals a change when the difference between the current cumulative sum and its historical minimum exceeds a decision threshold λ:

$$\textit{PH alarm:} \quad m_T - M_T > \lambda \tag{47}$$

The intuition is direct: $m_T - M_T$ measures the cumulative evidence that the signal has increased above its historical level. Under the null hypothesis (no change), this quantity fluctuates around zero. After an abrupt upward shift, the cumulative sum m_T begins increasing systematically while the minimum M_T remains anchored at its pre-shift level, causing the difference to grow until it exceeds λ.

7.3.2 Theoretical Properties

- **Average Run Length (ARL) under null**: The expected number of observations before a false alarm is $\text{ARL}_0 \approx e^{c\lambda}$ for a constant c depending on the signal variance. This grows exponentially with λ, providing strong control over false positives.
- **Average Detection Delay**: For an abrupt shift of magnitude Δ in the mean, the expected detection delay is approximately $\lambda/(\Delta - \nu)$ observations (for $\Delta > \nu$). Larger shifts are detected faster.
- **Per-observation complexity**: $O(1)$. The test updates two scalars (m_T, M_T) and one running mean per observation.

7.3.3 Role in Clause AI-6

The PH test is specifically tuned for two catastrophic drift signatures:

Signature 1: Adversarial injection spike. A jailbreak or prompt injection produces a step-function increase in drift-kl from $\sim$ 0.01–0.05 nats (normal) to $>$ 1.0 nats (adversarial output). The shift magnitude $\Delta \gg \nu$, producing detection within a small number of tokens.

Signature 2: Repetition loop collapse. When a model enters a repetition loop, its output entropy collapses toward zero and its drift-kl against a diverse baseline rises sharply. The PH test detects the onset of the loop—the point at which the cumulative evidence of elevated drift exceeds λ—before the loop becomes entrenched.

Signature 3: Gibberish generation. Hardware failure, weight corruption, or extreme quantization errors can cause the model to produce incoherent output with exploding entropy. This manifests as an abrupt, sustained increase in drift-kl that the PH test detects as a mean shift.

7.3.4 Calibrated Parameters

Table 19: Page-Hinkley Parameters for Clause AI-6

Parameter	Value	Rationale
Allowance ν	0.05 nats	Set above the normal jitter range (0.01–0.05); only shifts exceeding typical
Decision threshold λ	0.5 nats	Calibrated for $\text{ARL}_0 > 10{,}000$ observations (approx. 10 minutes at 1 obs/e
Min observations	30	Minimum observations before alarm is eligible; prevents warm-up artifacts

The choice of $\nu = 0.05$ nats is critical. The normal drift-kl jitter range under null conditions is 0.01–0.05 nats (Section 2, Table 2). Setting ν at the upper bound of this range ensures that only drift signals *exceeding* normal variation accumulate positive evidence in the PH cumulative sum. Normal fluctuations produce near-zero or negative increments, preventing false accumulation.

7.4 KSWIN (Kolmogorov-Smirnov Windowing)

KSWIN (Rabanser et al., 2019) uses the Kolmogorov-Smirnov (KS) two-sample test to compare the most recent window of data against a reference window. It provides the "second opinion" in the Clause AI-6 pipeline: when ADWIN signals a potential drift event, KSWIN provides an independent statistical confirmation with higher rigor before a compliance state transition is declared.

7.4.1 Algorithm Description

Definition 7.3 (KSWIN)**.** *KSWIN maintains two windows of drift-kl observations:*

- *A* reference window *R of the most recent n_R observations known to be in-distribution (initialized from the CoRIM construction phase).*
- *A* test window *T of the most recent n_T observations from production.*

At each evaluation point, the KS statistic is computed:

$$D_{KS} = \sup_x |F_R(x) - F_T(x)| \tag{48}$$

where F_R and F_T are the empirical cumulative distribution functions of the reference and test windows respectively.

A change is signaled when:

$$D_{KS} > c(\alpha_{KS}) \cdot \sqrt{\frac{n_R + n_T}{n_R \cdot n_T}} \tag{49}$$

where $c(\alpha_{KS})$ is the critical value of the KS distribution at significance level α_{KS}.

7.4.2 Theoretical Properties

- **Non-parametric**: The KS test makes no assumptions about the distribution of drift-kl values. It is sensitive to changes in any property of the distribution—mean, variance, shape, tails—not just the mean (which is all that ADWIN and PH test detect).
- **Consistency**: For any continuous distribution, the KS test is consistent: as $n_R, n_T \to \infty$, the power against any fixed alternative converges to 1.
- **False alarm control**: Under the null hypothesis, $\Pr[D_{\text{KS}} > c(\alpha_{\text{KS}}) \cdot \sqrt{(n_R + n_T)/(n_R n_T)}] = \alpha_{\text{KS}}$.
- **Computational complexity**: $O(n \log n)$ per evaluation (due to sorting), where $n = n_R + n_T$.

7.4.3 Why Shape Sensitivity Matters

ADWIN and PH detect changes in the *mean* of the drift-kl signal. But drift can manifest as a change in the *variance* or *shape* of the drift-kl distribution without a change in the mean. Consider:

- **Bimodal drift**: The model alternates between two operating regimes—one producing low drift-kl (≈ 0.02) and one producing high drift-kl (≈ 0.4). The mean drift-kl may remain within bounds, but the bimodal distribution indicates that the model is intermittently failing. ADWIN would not detect this; KSWIN's sensitivity to distributional shape would.
- **Variance explosion**: The model's drift-kl becomes increasingly volatile—sometimes very low, sometimes very high—without a systematic increase in the mean. This indicates instability that precedes a more severe failure. KSWIN detects the change in the spread of the distribution.

7.4.4 Triggered Evaluation (Not Always-On)

Due to its higher computational cost ($O(n \log n)$ vs. ADWIN's $O(\log W)$), KSWIN is *not* run on every observation. Instead, it is triggered by one of two conditions:

1. **ADWIN warning**: When ADWIN detects a window cut (Section 7.2), KSWIN is invoked to confirm the detection. If KSWIN confirms (D_{KS} exceeds the critical value), the drift event is declared with high confidence. If KSWIN does not confirm, the ADWIN detection is logged as a "soft warning" but does not trigger a compliance state transition.

2. **Periodic audit**: Every n_{audit} monitoring epochs (default: 100), KSWIN is invoked regardless of ADWIN state. This provides an independent, periodic check that catches distributional changes (shape, variance) that ADWIN's mean-based detection would miss.

7.4.5 Calibrated Parameters

Table 20: KSWIN Parameters for Clause AI-6

Parameter	Value	Rationale
Reference window n_R	200	Observations from CoRIM construction (known in-distribution)
Test window n_T	100	Most recent production observations
Significance α_{KS}	0.01	1% false alarm rate per evaluation; conservative for governance
Audit interval n_{audit}	100 epochs	Periodic evaluation regardless of ADWIN state

7.5 Algorithm Composition: The Detection Pipeline

The three algorithms are composed into a unified detection pipeline with formally characterized properties. The pipeline operates on the streaming drift-kl signal from the multi-scale monitoring windows (Section 6.2).

7.5.1 Pipeline Architecture

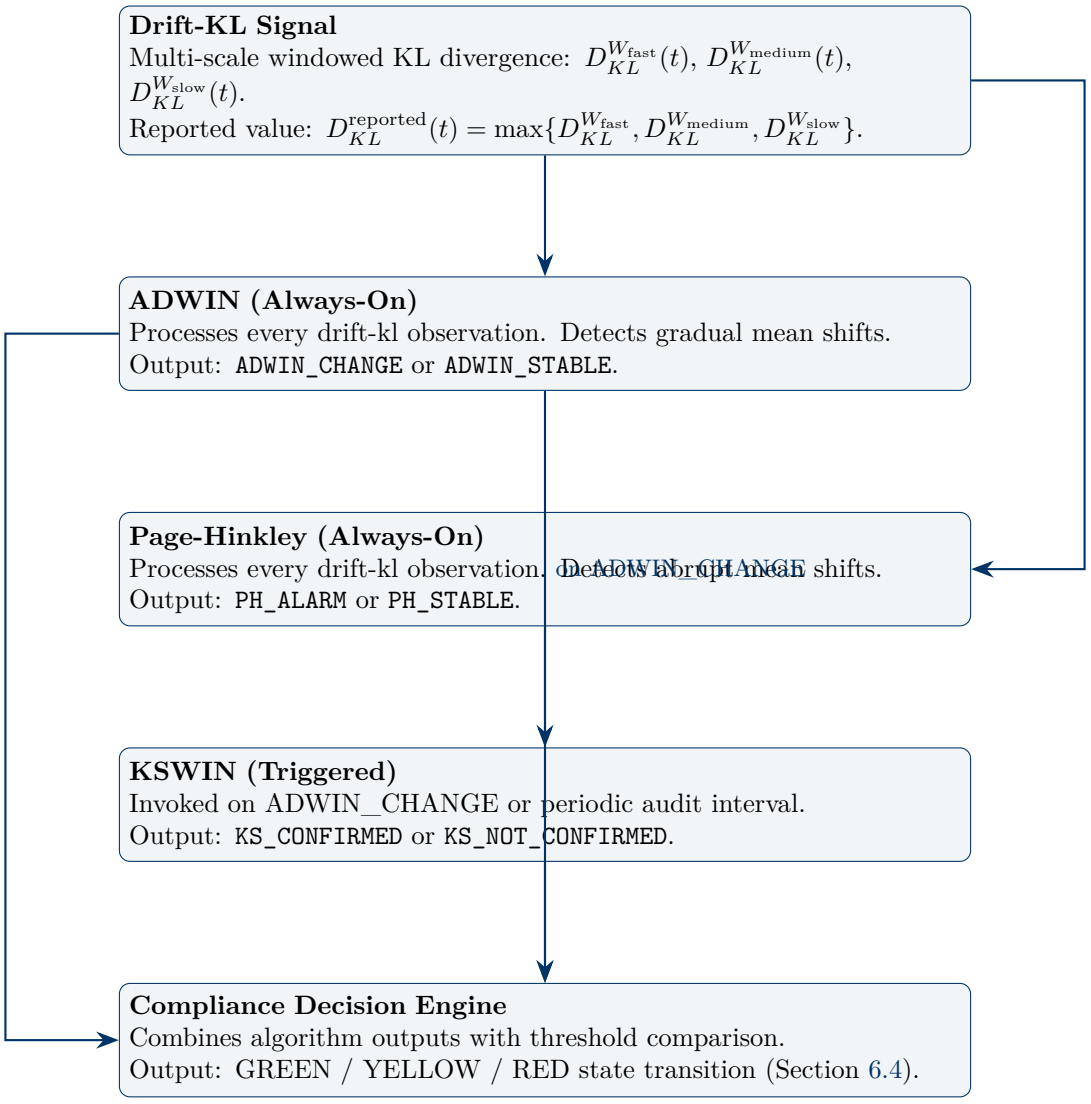

Figure 3: Clause AI-6 Detection Pipeline: Algorithm Composition

7.5.2 Decision Logic

The Compliance Decision Engine combines the three algorithm outputs with the direct threshold comparison to produce state transitions:

Detection Pipeline Decision Logic

The following rules govern the mapping from algorithm outputs to compliance state transitions. Rules are evaluated in priority order (highest first).

R1: PH_ALARM ⇒ immediate RED. A Page-Hinkley alarm indicates catastrophic drift. The compliance state transitions to RED immediately, bypassing KSWIN confirmation. Rationale: the PH test has its own false alarm control ($\text{ARL}_0 > 10{,}000$); requiring additional confirmation would delay response to catastrophic events.

R2: $D_{KL}^{\textbf{reported}}(t) > D_{\max}$ **⇒ RED.** Direct threshold exceedance triggers RED regardless of algorithm state (transition T3, Section 6.4).

R3: ADWIN_CHANGE + KS_CONFIRMED ⇒ YELLOW or RED. ADWIN detects a mean shift, confirmed by KSWIN's distributional test. If the current $D_{KL}^{\text{reported}}(t) > D_{\text{yellow}}$, transition to YELLOW (T1). If $D_{KL}^{\text{reported}}(t) > D_{\max}$, transition to RED (T3).

R4: ADWIN_CHANGE + KS_NOT_CONFIRMED ⇒ soft warning. ADWIN detected a change but KSWIN did not confirm. Logged as a monitoring event but does not trigger a compliance state transition. The sampling rate for drift-kl computation **SHOULD** be increased (see Section 9) for the next n_{audit} epochs to capture additional evidence.

R5: $D_{KL}^{\textbf{reported}}(t) > D_{\textbf{yellow}}$ **⇒ YELLOW.** Direct threshold exceedance of the warning level triggers YELLOW regardless of algorithm state (transition T1).

R6: All stable, $D_{KL}^{\textbf{reported}}(t) \leq D_{\textbf{yellow}}$ **⇒ GREEN.** Normal operation. If recovering from YELLOW, the N_{recovery} condition (transition T2) must be satisfied.

7.5.3 Composite Properties

The composed pipeline inherits formal properties from its constituent algorithms:

Proposition 7.3 (Pipeline False Alarm Rate)**.** *Under the null hypothesis (no drift), the probability of the pipeline producing a RED state transition at any given monitoring epoch is bounded by:*

$$\Pr[RED \mid H_0] \leq \Pr[PH\_ALARM \mid H_0] + \Pr[D_{KL}^{reported} > D_{\max} \mid H_0] \tag{50}$$

The first term is controlled by the PH parameters ($ARL_0 > 10{,}000$, corresponding to $\Pr \lesssim 10^{-4}$ per epoch). The second term is controlled by the conformal calibration of $D_{\max}$ ($\Pr \leq \alpha_{FP}$, Table 11). For typical configurations, the combined false RED rate is $< 0.01\%$ per monitoring epoch.

Proposition 7.4 (Pipeline Detection Delay)**.** *For an abrupt drift event of magnitude Δ (nats):*

- *If $\Delta > 0.5$ nats (catastrophic): PH detects within $\lambda/(\Delta - \nu) \approx 0.5/(0.5 - 0.05) \approx 1.1$ observations. Detection is near-instantaneous.*
- *If $0.1 < \Delta < 0.5$ nats (moderate): ADWIN detects within $O(1/\Delta^2 \cdot \log(1/\delta))$ observations, confirmed by KSWIN. Detection latency: ∼10–100 observations depending on magnitude.*
- *If $\Delta < 0.1$ nats (gradual): The slow window W_{slow} accumulates statistical evidence. Detection latency: ∼1,000–10,000 observations.*

7.6 Handling Algorithm State Across Sessions

LLM deployments serve multiple concurrent users across overlapping sessions. The detection algorithms must handle the distinction between *per-session* drift (a single user's interaction is drifting) and *global* drift (the model's overall output distribution is shifting).

7.6.1 Per-Session Monitoring

Each active session maintains its own W_{fast} window and PH accumulator. This enables detection of adversarial prompt injection (Section 2.7) and coherence collapse (Section 2.1) within individual sessions. Per-session alarms trigger session-level interventions (e.g., safe mode activation for that session) without affecting global compliance state.

7.6.2 Global Monitoring

The W_{medium} and W_{slow} windows aggregate drift-kl observations across all active sessions. ADWIN and KSWIN operate on the global stream. Global alarms trigger system-level compliance state transitions affecting the MAI-1 attestation payload.

7.6.3 Aggregation Protocol

Session-to-Global Aggregation

The global drift-kl observation at monitoring epoch t is the weighted average of per-session drift-kl values:

$$D_{KL}^{\text{global}}(t) = \frac{1}{\sum_s n_s(t)} \sum_{s \in \mathcal{S}_{\text{active}}} n_s(t) \cdot D_{KL}^{s}(t) \tag{51}$$

where $\mathcal{S}_{\text{active}}$ is the set of active sessions, $n_s(t)$ is the number of tokens generated in session s during epoch t, and $D_{KL}^{s}(t)$ is the per-session drift-kl for session s.
This token-weighted average ensures that sessions generating more output (and thus providing more statistical evidence) contribute proportionally to the global signal.

7.7 Feasibility Analysis

The detection pipeline must operate within the computational budget of a production LLM system generating thousands of tokens per second.

Table 21: Detection Pipeline Computational Feasibility

Component	Per-Observation	Frequency	Amortized Cost
ADWIN update	$O(\log W) \approx 1\,\mu$s	Every observation	$1\,\mu$s per obs.
PH update	$O(1) \approx 0.1\,\mu$s	Every observation	$0.1\,\mu$s per obs.
KSWIN evaluation	$O(n \log n) \approx 50\,\mu$s	On ADWIN trigger or every 100 epochs	$0.5\,\mu$s per obs. (amortized)
Per-session PH	$O(1) \approx 0.1\,\mu$s	Every observation, per session	$0.1\,\mu$s$\times$ concurrent sessions
Total pipeline			$\approx$ 2–5 μs per observation

For a model generating 10,000 tokens per second with a monitoring observation rate of 1 observation per 100 tokens (the 1% sampling rate from Section 9), the pipeline processes 100 observations per second. At 5 μs per observation, the total pipeline cost is 500 μs/s—0.05% of

available compute. This is well within the 1–2% latency budget, leaving the majority of the budget for the drift-kl computation itself (Section 9).

Honest Framing

What the detection pipeline provides: Formal false alarm control, sensitivity to both gradual and abrupt drift, shape-aware distributional testing, and computational feasibility at production token generation rates. The three-algorithm composition ensures that no single algorithm's blind spot creates a detection gap.

What the detection pipeline does not provide: Guaranteed detection of all drift events within a fixed time bound. Detection delay depends on drift magnitude: small, gradual drifts may require hundreds of thousands of observations for reliable detection. The pipeline is designed for statistical reliability, not for worst-case latency guarantees. Safety-critical deployments requiring worst-case bounds should additionally implement the direct threshold comparison (Rule R2) on the fast window W_{fast}, accepting a higher false positive rate in exchange for guaranteed detection latency.

8 Conformal Prediction Integration

The drift-kl invariant (Section 6) and the online detection algorithms (Section 7) provide the primary monitoring infrastructure for Clause AI-6. But both operate on the *aggregate* output distribution—they detect shifts in the population-level behavior of the model. Neither provides per-prediction uncertainty quantification: for any individual model output, the system cannot state, using drift-kl alone, how confident it is that the output is within the model's validated operating envelope.

Conformal prediction (CP) closes this gap. It provides *distribution-free, finite-sample coverage guarantees* for individual predictions, requiring only the assumption that the calibration data and the production data are exchangeable (i.e., drawn from the same distribution in any order). Crucially, this exchangeability assumption is precisely what drift violates—making CP both a per-prediction uncertainty quantifier and an independent drift detector. When the model drifts, the conformal coverage degrades, providing a signal that is complementary to and independent of drift-kl.

This section develops the conformal prediction framework for Clause AI-6: the theoretical foundations, the specific CP methodology adapted for autoregressive language models, the coverage-drift linkage that transforms CP into a drift detector, the adaptive extensions for non-stationary environments, and the connection to commercial insurability that makes conformal prediction the bridge between governance infrastructure and financial risk transfer.

8.1 Conformal Prediction: Theoretical Foundations

8.1.1 The Coverage Guarantee

Definition 8.1 (Conformal Prediction)**.** *Let $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be a calibration set of input-output pairs, and let (X_{n+1}, Y_{n+1}) be a new test point. A conformal predictor constructs a prediction set $\mathcal{C}(X_{n+1}) \subseteq \mathcal{Y}$ such that:*

$$\Pr[Y_{n+1} \in \mathcal{C}(X_{n+1})] \geq 1 - \alpha \tag{52}$$

where $\alpha \in (0, 1)$ is the user-specified miscoverage rate, and the probability is over the joint distribution of the calibration set and the test point.

The guarantee in Equation 52 is *marginal*: it holds on average over the randomness of both the calibration set and the test point. It is not conditional on a specific input X_{n+1}—the

coverage may be higher for some inputs and lower for others, but the average over the input distribution is guaranteed.

8.1.2 The Exchangeability Assumption

Definition 8.2 (Exchangeability)**.** *A sequence of random variables $(Z_1, Z_2, \ldots, Z_n)$ is exchangeable if for every permutation π of $\{1, \ldots, n\}$:*

$$(Z_1, Z_2, \ldots, Z_n) \overset{d}{=} (Z_{\pi(1)}, Z_{\pi(2)}, \ldots, Z_{\pi(n)}) \tag{53}$$

That is, the joint distribution is invariant under permutation of the indices.

Exchangeability is strictly weaker than the i.i.d. assumption: i.i.d. random variables are exchangeable, but exchangeable random variables need not be independent or identically distributed. This makes conformal prediction applicable to a broader class of data-generating processes than methods requiring i.i.d. assumptions.

Theorem 8.1 (Conformal Coverage (Vovk et al., 2005))**.** *If $(Z_1, \ldots, Z_n, Z_{n+1})$ are exchangeable, and the non-conformity scores $s_1, \ldots, s_n, s_{n+1}$ are computed using a fixed non-conformity measure, then:*

$$\Pr[s_{n+1} \leq \hat{q}_{1-\alpha}] \geq 1 - \alpha \tag{54}$$

where $\hat{q}_{1-\alpha}$ is the $\lceil (1-\alpha)(n+1) \rceil / n$ quantile of the calibration scores $\{s_1, \ldots, s_n\}$.

This theorem is the foundation of all conformal prediction methods. It requires no assumptions about the distribution of the data, the form of the model, or the relationship between inputs and outputs. The only assumption is exchangeability.

8.1.3 Non-Conformity Scores for Language Models

The non-conformity score $s(x, y)$ measures how "unusual" the output y is for input x according to the model. For autoregressive language models, several non-conformity scores are applicable:

Score 1: Negative log-likelihood.

$$s_{\text{NLL}}(x, y) = -\log P_\theta(y \mid x) = -\sum_{t=1}^{|y|} \log P_\theta(y_t \mid x, y_{<t}) \tag{55}$$

This is the most natural score for generative models: it measures the total surprisal of the output sequence under the model's own probability distribution. Higher NLL indicates a less likely (more "non-conforming") output. The corresponding prediction set $\mathcal{C}_\alpha(x) = \{y : s_{\text{NLL}}(x, y) \leq \hat{q}_{1-\alpha}\}$ contains all output sequences whose likelihood exceeds the calibrated threshold.

Score 2: Per-token maximum surprisal.

$$s_{\max}(x, y) = \max_{1 \leq t \leq |y|} \left(-\log P_\theta(y_t \mid x, y_{<t}) \right) \tag{56}$$

This score captures the single most surprising token in the sequence. It is more sensitive to *local* anomalies (a single hallucinated entity name, a sudden language switch) than the NLL score, which averages over the full sequence and can mask local spikes in a long, otherwise-normal output.

Score 3: Drift-weighted surprisal.

$$s_{\text{drift}}(x, y) = \sum_{t=1}^{|y|} w_t \cdot (-\log P_\theta(y_t \mid x, y_{<t})), \quad w_t = 1 + \gamma \cdot D_{KL}^{\text{local}}(t) \tag{57}$$

where $D_{KL}^{\text{local}}(t)$ is the per-token contribution to drift-kl at position t, and $\gamma > 0$ is a sensitivity parameter. This score up-weights tokens that contribute disproportionately to drift, making the conformal prediction set sensitive to the same distributional shifts that drift-kl detects. This creates a tight coupling between the drift invariant and the per-prediction uncertainty quantification.

Non-Conformity Score Selection

MAI-1 conformant deployments **MUST** compute at least one non-conformity score from the set $\{s_{\text{NLL}}, s_{\max}, s_{\text{drift}}\}$ for conformal prediction integration. The score selection and calibration methodology **MUST** be documented in the vendor evidence package. High-risk deployments **SHOULD** compute both s_{NLL} (for sequence-level uncertainty) and $s_{\max}$ (for token-level anomaly detection).

8.2 Split Conformal Prediction for LLMs

The standard conformal prediction framework (Theorem 8.1) requires re-fitting the model for each candidate output, which is computationally prohibitive for LLMs. Split conformal prediction (Papadopoulos et al., 2002; Lei et al., 2018) resolves this by separating the calibration phase from the prediction phase.

8.2.1 Calibration Phase (Offline)

During CoRIM construction (Section 5.5), the conformal calibration proceeds as follows:

1. The validation set $\mathcal{D}_{\text{val}}$ is partitioned into a training set $\mathcal{D}_{\text{train}}$ (used for model development) and a calibration set $\mathcal{D}_{\text{cal}}$ (reserved exclusively for conformal calibration). The calibration set **MUST** not have been used during model training, fine-tuning, or hyperparameter selection.

2. For each calibration example $(x_i, y_i) \in \mathcal{D}_{\text{cal}}$, the non-conformity score $s_i = s(x_i, y_i)$ is computed using the frozen production model.

3. The calibration scores $\{s_1, \ldots, s_{|\mathcal{D}_{\text{cal}}|}\}$ are sorted in ascending order.

4. The conformal threshold is set as the empirical quantile:

$$\hat{q}_{1-\alpha} = s_{(\lceil (1-\alpha)(|\mathcal{D}_{\text{cal}}|+1) \rceil)} \tag{58}$$

5. The threshold $\hat{q}_{1-\alpha}$, the score type, and the calibration metadata are serialized into the CoRIM artifact as companion data to the distribution reference.

8.2.2 Prediction Phase (Online)

At inference time, for each model output $(x_{\text{new}}, y_{\text{new}})$:

1. Compute the non-conformity score $s_{\text{new}} = s(x_{\text{new}}, y_{\text{new}})$.

2. Compare against the conformal threshold: $s_{\text{new}} \leq \hat{q}_{1-\alpha}$?

3. If yes: the output is *conforming*—it falls within the model's calibrated prediction set. Report `conformal-status = CONFORMING` in the MAI-1 payload.

4. If no: the output is *non-conforming*—it is more surprising than $(1 - \alpha) \times 100\%$ of the calibration outputs. Report `conformal-status = NON_CONFORMING`.

The computational cost of the prediction phase is negligible: a single scalar comparison per output. The non-conformity score s_{NLL} is already computed as part of the model's forward pass (it is the cross-entropy loss); $s_{\max}$ requires tracking the maximum per-token surprisal during generation, which is a single running-maximum operation.

8.3 The Exchangeability-Drift Linkage

The conformal coverage guarantee (Theorem 8.1) holds if and only if the calibration data and the production data are exchangeable. Distribution drift, by definition, violates exchangeability: if $P_t \neq P_0$, then the production data is drawn from a different distribution than the calibration data, and the joint sequence is no longer permutation-invariant.

This creates a powerful linkage: **when drift occurs, conformal coverage degrades.**

8.3.1 Formal Statement

Proposition 8.2 (Coverage Degradation Under Drift)**.** *Let P_0 be the calibration distribution and P_t the production distribution at time t. If $D_{KL}(P_t \| P_0) > 0$, the empirical coverage of the conformal predictor satisfies:*

$$\text{Coverage}(t) = \Pr_{(x,y)\sim P_t}[s(x, y) \leq \hat{q}_{1-\alpha}] \leq (1 - \alpha) + \epsilon(D_{KL}) \tag{59}$$

where $\epsilon(D_{KL})$ is a correction term that can be positive or negative depending on the direction of drift. Under typical drift scenarios (the production distribution has heavier tails or shifted modes relative to calibration), the correction is negative:

$$\text{Coverage}(t) < 1 - \alpha \tag{60}$$

The coverage drops below the guaranteed level.

Remark 8.1. *The precise relationship between $D_{KL}(P_t \| P_0)$ and coverage degradation depends on how the drift affects the non-conformity score distribution. Drift that increases the model's uncertainty (heavier tails, higher entropy) tends to decrease coverage (more outputs exceed the threshold). Drift that decreases uncertainty (mode collapse, repetition) tends to increase coverage (fewer outputs exceed the threshold, but the outputs themselves are degenerate). Both directions are governance-relevant: under-coverage indicates the model is producing surprising outputs; over-coverage may indicate mode collapse.*

8.3.2 Coverage Monitoring as Drift Detection

The coverage degradation result transforms conformal prediction from a per-prediction uncertainty quantifier into an *independent drift detector*. The monitoring procedure:

1. Track the empirical coverage rate over a sliding window of N_{CP} recent predictions:

$$\hat{C}(t) = \frac{1}{N_{\text{CP}}} \sum_{i=t-N_{\text{CP}}+1}^{t} \mathbf{1}[s_i \leq \hat{q}_{1-\alpha}] \tag{61}$$

2. Under the null hypothesis (no drift), $\hat{C}(t)$ fluctuates around $(1 - \alpha)$ with standard deviation $\sigma_C = \sqrt{\alpha(1 - \alpha)/N_{\text{CP}}}$.

3. A coverage violation is signaled when the empirical coverage drops below a lower bound:

$$\hat{C}(t) < (1 - \alpha) - z_\beta \cdot \sigma_C \tag{62}$$

where z_β is the β-quantile of the standard normal distribution, controlling the false alarm rate of the coverage monitor.

The coverage monitor is independent of drift-kl: it uses different data (per-prediction conformity rather than aggregate distributional statistics), different statistical tests (binomial coverage rate rather than KL divergence), and detects different drift types (any drift that violates exchangeability, including covariate shift that drift-kl may miss; Table 4).

Table 22: Conformal Coverage Monitoring Parameters

Parameter	**Value**	**Rationale**
Miscoverage rate α	0.10	90% target coverage; standard in conformal literature
Coverage window N_{CP}	500	Provides $\sigma_C \approx 0.013$; detects 3% coverage drops at $z_\beta = 2.33$
Alarm quantile z_β	2.33	1% false alarm rate (one-sided)
Coverage alarm threshold	0.87	$(1 - \alpha) - z_\beta \cdot \sigma_C = 0.90 - 2.33 \times 0.013$

8.4 Prediction Set Width as a Utility Metric

The conformal prediction set $\mathcal{C}_\alpha(x)$ contains all outputs whose non-conformity score does not exceed the threshold. The *width* (or cardinality, or volume) of this set provides a utility metric that is independent of coverage.

8.4.1 Definition

For the NLL-based non-conformity score, the prediction set width can be expressed as the effective number of "plausible" outputs:

$$|\mathcal{C}_\alpha(x)| \approx \exp(\hat{q}_{1-\alpha}) \tag{63}$$

More precisely, the set width at the token level is the expected size of the set of tokens with per-token NLL below the per-token threshold:

$$w_t(x) = |\{y \in \mathcal{V} : -\log P_\theta(y \mid x, y_{<t}) \leq q_t\}| \tag{64}$$

8.4.2 Interpretation

- **Narrow sets** (w_t small): The model is confident about the next token. Few alternatives are plausible. The prediction is precise.
- **Wide sets** (w_t large): The model is uncertain. Many tokens are roughly equally plausible. The prediction is imprecise.
- **Expanding sets over time**: If w_t increases systematically across the monitoring window, the model is becoming less confident about its predictions—a possible precursor to coherence collapse or drift.
- **Contracting sets over time**: If w_t decreases systematically, the model is becoming more confident—possibly indicating mode collapse or inertial drift (the model is "locking in" to a narrow output pattern).

The set width trend is reported as a companion metric in the MAI-1 Layer 2 payload:

Listing 2: MAI-1 Conformal Prediction Extension Fields

```
"conformal": {
    "score-type": "nll",           // or "max-surprisal", "drift-
       weighted"
    "threshold": 4.82,             // calibrated quantile
    "target-coverage": 0.90,
    "empirical-coverage": 0.88,    // sliding window estimate
    "coverage-status": "WARNING",  // NOMINAL, WARNING, VIOLATION
    "mean-set-width": 127.3,       // average token-level set width
    "set-width-trend": 0.034,      // positive = expanding (less
       confident)
    "calibration-corim": "urn:corim:ai6-cp-cal-v1"
}
```

8.5 Adaptive Conformal Prediction for Non-Stationary Environments

Standard split conformal prediction assumes exchangeability between calibration and production data. In the presence of drift, this assumption is violated and the coverage guarantee degrades (Proposition 8.2). Adaptive Conformal Inference (ACI; Gibbs & Candès, 2021) extends conformal prediction to non-stationary environments by dynamically adjusting the conformal threshold based on recent coverage performance.

8.5.1 The ACI Algorithm

Definition 8.3 (Adaptive Conformal Inference (Gibbs & Candès, 2021))**.** *ACI maintains an adaptive miscoverage level α_t that evolves over time:*

$$\alpha_{t+1} = \alpha_t + \eta\,(\alpha - \mathbf{1}[Y_t \notin \mathcal{C}_{\alpha_t}(X_t)]) \tag{65}$$

where:

- *α is the target miscoverage rate (e.g., 0.10);*
- *$\eta > 0$ is the learning rate controlling the adaptation speed;*
- *$\mathbf{1}[Y_t \notin \mathcal{C}_{\alpha_t}(X_t)]$ is the indicator of a miscoverage event at time t;*
- *$\mathcal{C}_{\alpha_t}(X_t)$ is the prediction set constructed using the current adaptive level α_t.*

The conformal threshold is then set as the $\lceil (1-\alpha_t)(n+1) \rceil / n$ quantile of the calibration scores, which tightens or loosens as α_t adapts.

8.5.2 Guarantee

Theorem 8.3 (ACI Long-Run Coverage (Gibbs & Candès, 2021))**.** *For any sequence of distributions (possibly adversarially chosen), ACI achieves:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbf{1}[Y_t \notin \mathcal{C}_{\alpha_t}(X_t)] \xrightarrow{T\to\infty} \alpha \tag{66}$$

That is, the long-run average miscoverage rate converges to the target α, regardless of how the underlying distribution changes over time.

This is a remarkable guarantee: ACI maintains the target coverage rate *even under arbitrary drift*, by automatically adjusting the prediction set width. When drift causes increased miscoverage, α_t decreases, the threshold tightens, and the prediction sets widen to restore coverage. When the model stabilizes, α_t increases toward α, and the sets contract.

8.5.3 The Governance Trade-off

ACI's guarantee comes with a cost: the prediction set width varies over time. Under drift, the sets widen (to maintain coverage), which reduces the *informativeness* of the prediction. In the extreme—severe drift where the model's output distribution has no overlap with the calibration distribution—ACI would expand the prediction set to include the entire vocabulary, maintaining 90% coverage but providing no useful uncertainty information.

This trade-off is itself a governance signal:

- **Set width expansion without drift-kl alarm**: ACI is detecting subtle drift that the aggregate drift-kl metric has not yet captured (possibly covariate shift). This is an early warning.
- **Set width expansion with drift-kl alarm**: Both metrics agree; the drift is confirmed and the model's per-prediction reliability has degraded.
- **Set width contraction without drift-kl alarm**: The model has become more confident (possibly entering a mode-collapse regime). The prediction sets are narrower but may be narrower than justified by the model's actual capability.

8.5.4 ACI Parameters for Clause AI-6

Table 23: Adaptive Conformal Inference Parameters

Parameter	Value	Rationale
Target miscoverage α	0.10	Consistent with static CP target
Learning rate η	0.01	Slow adaptation; prevents oscillation from noisy individual outputs
α_t bounds	$[\alpha/5,\ 5\alpha]$	Clamped to $[0.02, 0.50]$; prevents degenerate thresholds

The learning rate $\eta = 0.01$ is deliberately conservative. A single miscoverage event adjusts α_t by $0.01 \times (0.10 - 1) = -0.009$—a small tightening. This prevents individual anomalous outputs from causing large, oscillatory swings in the prediction set width. Sustained drift, on the other hand, produces a consistent stream of miscoverage events that accumulate over hundreds of predictions, driving α_t steadily downward and the prediction sets steadily wider.

8.6 The Insurance Connection: Conformal Prediction and Commercial Insurability

The conformal prediction framework provides the mathematical bridge between the Auburn Governance Stack's technical monitoring infrastructure and the commercial insurance market's requirement for quantifiable failure probabilities.

8.6.1 The Insurability Problem

For an insurer to underwrite an AI system, the insurer must be able to:

1. Estimate the probability of a covered loss event (actuarial pricing).
2. Bound the maximum expected loss from a single event (policy sizing).
3. Verify that the insured system maintains the operational conditions assumed in pricing (ongoing compliance).

Traditional insurance (auto, health, property) builds actuarial models from decades of claims history. AI systems have no such history. The conformal prediction framework resolves this by providing *forward-looking, distribution-free failure probability bounds* without requiring historical claims data.

8.6.2 CP-Bounded Failure Probability

The conformal coverage guarantee (Equation 52) directly provides a failure probability bound:

$$\Pr[\text{output non-conforming}] \leq \alpha \tag{67}$$

For $\alpha = 0.10$, no more than 10% of model outputs are expected to fall outside the calibrated prediction set under normal operating conditions. If the insurer defines a "covered loss event" as the occurrence of a non-conforming output that causes downstream harm, the conformal bound provides a first-order estimate of the event frequency.

8.6.3 The Munich Re *aiSure* Connection

Munich Re's *aiSure* product provides performance guarantees for AI systems, backed by reinsurance. The *aiSure* framework requires:

1. A quantitative performance metric with a certified bound.
2. A monitoring mechanism that verifies ongoing compliance with the bound.
3. A remediation protocol when compliance is violated.

Clause AI-6's conformal prediction integration maps directly to these requirements:

Table 24: Clause AI-6 to Munich Re *aiSure* Requirement Mapping

aiSure Requirement	**Clause AI-6 Component**	**Evidence**
Quantitative performance metric	Conformal coverage rate $(1 - \alpha)$	Distribution-free bound; no distributional assumptions
Certified bound	$\hat{q}_{1-\alpha}$ threshold sealed in CoRIM	Cryptographically signed; tamper-evident; SCITT-registered
Monitoring mechanism	Coverage tracking (Equation 61) + ACI (Section 8.5)	Continuous; automated; independent of model vendor
Remediation protocol	Compliance state machine (Section 6.4)	Binary determination; automated intervention on RED
Drift detection	Exchangeability violation → coverage degradation	Independent of drift-kl; complementary signal

8.6.4 Actuarial Integration

The conformal framework enables the following actuarial workflow:

1. **Pricing**: The insurer sets the premium based on the conformal miscoverage rate α, the expected severity of non-conforming events (estimated from the calibration set's worst-case non-conformity scores), and the deployment context risk multiplier.
2. **Policy terms**: The policy specifies that coverage is conditional on the insured system maintaining:

- Drift-kl compliance (GREEN or YELLOW state).
- Conformal coverage above the alarm threshold (Equation 62).
- Valid, unexpired CoRIM with registered Sentinel Evidence Package for any baseline transitions.

3. **Claims assessment**: When a loss event occurs, the insurer retrieves the MAI-1 attestation payload for the relevant time period and verifies:

   - Was the system in drift-kl compliance at the time of the event?
   - Was the conformal coverage within bounds?
   - Was the specific output flagged as non-conforming by the conformal predictor?

   If the system was out of compliance at the time of the event, the claim may be denied or the payout reduced, analogous to a building fire claim being adjusted when the building's fire suppression system was non-functional.

4. **Reserve estimation**: The conformal coverage rate provides a directly usable input for loss reserve calculations:

$$\text{Expected annual losses} \leq \alpha \times N_{\text{annual}} \times \bar{L} \tag{68}$$

   where N_{annual} is the expected number of model outputs per year and $\bar{L}$ is the average loss per non-conforming event.

8.6.5 The Commercial Significance

The conformal prediction integration transforms Clause AI-6 from a technical monitoring specification into a *commercially insurable* framework. Without CP, the drift-kl metric provides aggregate distributional monitoring but cannot bound the probability of individual failure events. With CP, the insurer has:

- A *distribution-free* failure probability bound (no parametric modeling assumptions).
- A *calibrated* bound (derived from the model's own validation data, not from an external actuarial model).
- A *verifiable* bound (the conformal threshold is sealed in the CoRIM and can be independently verified by any party).
- An *adaptive* bound (ACI maintains coverage under drift, with set width expansion providing an early warning signal).

This combination of properties is, as of 2026, unique to the Auburn Governance Stack. No other AI governance framework provides distribution-free, cryptographically verifiable, per-prediction failure probability bounds that are compatible with commercial insurance underwriting.

Honest Framing

What conformal prediction provides: Distribution-free, finite-sample coverage guarantees for individual model outputs; an independent drift detection mechanism via exchangeability violation; a bridge to commercial insurability through quantifiable failure probability bounds; and adaptive coverage maintenance under non-stationary conditions.
What conformal prediction does not provide: Conditional coverage guarantees (coverage is marginal over the input distribution, not guaranteed for specific input types); detection of the *cause* of drift (CP detects that exchangeability is violated but does not identify which factor of the joint distribution has changed); or protection against adversarial attacks specifically targeting the conformal predictor (an adversary who knows the threshold can craft outputs that are just barely conforming while being harmful). These limitations are mitigated by the complementary monitoring architecture: drift-kl provides aggregate detection, the detection pipeline (Section 7) provides causal signatures, and the synthetic baseline (Section 5.1.3) provides safety-specific detection.

9 Computational Tractability: Inference-Time Optimization

The preceding sections have specified the mathematical formalization (Section 6), the online detection algorithms (Section 7), and the conformal prediction integration (Section 8) for Clause AI-6. Each component is theoretically sound and governance-ready. But none of this matters if the monitoring infrastructure imposes latency or compute overhead that degrades the model's production performance. An attestation framework that slows inference by 10% will not be adopted—regardless of its governance value.

This section provides the complete computational optimization strategy for Clause AI-6 monitoring, demonstrating that the full monitoring suite can operate within a 1–2% latency overhead budget at production inference rates. The strategy employs four complementary techniques: logit projection, Top-K truncation, asynchronous sampling, and variational surrogate models. Each technique is formally characterized with respect to its approximation error, latency contribution, and interaction with the drift-kl accuracy requirements.

9.1 Latency Budget

The latency budget establishes the hard constraint: the total additional time per token attributable to Clause AI-6 monitoring must not exceed the allocated overhead.

9.1.1 Production Inference Baseline

Modern LLM inference operates at the following approximate latencies per output token:

Table 25: Production Inference Latency Baselines (Per Output Token)

Model Scale	Hardware	Latency/Token	1% Budget
7B (INT4)	Single A100	∼5 ms	50 μs
13B (INT8)	Single A100	∼10 ms	100 μs
70B (FP16)	4× A100	∼30 ms	300 μs
70B (INT4)	Single H100	∼15 ms	150 μs
405B (FP16)	8× H100	∼50 ms	500 μs

The 1% budget column establishes the maximum per-token latency contribution for Clause AI-6 monitoring at the most conservative overhead target. The 2% budget doubles these values.

All optimization techniques in this section are designed to fit within the 1% budget; deployments that can tolerate 2% overhead gain additional monitoring resolution.

9.1.2 Latency Budget Allocation

Table 26: Clause AI-6 Latency Budget Allocation (1% Target, 70B FP16 Baseline)

Component	Budget	Notes
Drift-KL computation	$\leq 150\,\mu$s	50% of budget; primary measurement
Conformal score computation	$\leq 30\,\mu$s	10% of budget; scalar comparison
Detection pipeline (ADWIN + PH)	$\leq 5\,\mu$s	$< 2\%$ of budget; negligible
Hidden state monitoring (surrogate)	$\leq 30\,\mu$s	10% of budget; single MLP forward pass
Overhead (aggregation, logging, state)	$\leq 85\,\mu$s	28% of budget; system margin
Total	$\leq 300\,\mu$s	1% of 30 ms baseline

The budget allocation reveals that the primary optimization challenge is the drift-KL computation itself. The detection algorithms (Section 7) are negligible. The conformal score (Section 8) is essentially free when using NLL-based non-conformity scores (the NLL is already computed during generation). The hidden state surrogate (Section 4.3.3) is a single small MLP forward pass. The drift-KL computation—computing the divergence between the production token distribution and the CoRIM reference—is where the optimization effort must be concentrated.

9.2 Technique 1: Logit Projection via Johnson-Lindenstrauss

The first optimization technique reduces the dimensionality of the hidden state space for hidden-state drift monitoring, using the Johnson-Lindenstrauss (JL) lemma to provide formal guarantees on the distortion introduced by the projection.

9.2.1 The Dimensionality Problem

Hidden state drift monitoring (Section 4.2.2) operates on vectors of dimension $d = 4{,}096$ to 12,288. Computing KL divergence between distributions in these spaces requires either the kNN estimator ($O(N^2)$ in high dimensions), the Donsker-Varadhan drift head (a trained MLP), or the variational surrogate. The kNN estimator is too slow for real-time use. The drift head and surrogate operate on the full-dimensional hidden state, requiring matrix multiplications of size $d \times h$ where h is the hidden layer width of the auxiliary network.

Reducing the dimensionality of the hidden state from d to $d' \ll d$ before passing it to the monitoring infrastructure reduces the computational cost of all downstream operations proportionally.

9.2.2 Johnson-Lindenstrauss Projection

Theorem 9.1 (Johnson-Lindenstrauss Lemma (1984))**.** *For any $\epsilon \in (0, 1)$ and any set of n points in $\mathbb{R}^d$, there exists a linear map $\Phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ with $d' = O(\epsilon^{-2} \log n)$ such that for all pairs of points x_i, x_j:*

$$(1 - \epsilon)\|x_i - x_j\|^2 \leq \|\Phi(x_i) - \Phi(x_j)\|^2 \leq (1 + \epsilon)\|x_i - x_j\|^2 \tag{69}$$

The map Φ can be constructed as a random matrix with entries drawn from $\mathcal{N}(0, 1/d')$.

The JL lemma guarantees that pairwise distances are preserved up to a multiplicative factor of $(1 \pm \epsilon)$ in the projected space. Since both the kNN estimator and the Donsker-Varadhan bound depend on distances between hidden states, the projected estimates inherit the JL distortion bound.

9.2.3 Projection Parameters for Clause AI-6

Table 27: Johnson-Lindenstrauss Projection Parameters

Original d	ϵ	n **(reference bank)**	**Target** d'	**Speedup**
4,096 (7B)	0.3	100,000	128	32×
8,192 (70B)	0.3	100,000	128	64×
12,288 (405B)	0.3	100,000	128	96×

A target dimension of $d' = 128$ with distortion $\epsilon = 0.3$ (30% maximum distance distortion) provides a substantial speedup while maintaining sufficient fidelity for drift detection. The 30% distortion is acceptable for governance purposes: the question is not "what is the exact KL divergence?" but "has drift exceeded a threshold?" A 30% distortion in the distance metric translates to a bounded error in the divergence estimate that is absorbed by the governance risk margin Δ_{gov} (Section 6.3).

9.2.4 Implementation

The projection matrix $\Phi \in \mathbb{R}^{d' \times d}$ is constructed once during CoRIM construction, serialized into the CoRIM artifact (alongside the distribution reference), and loaded at deployment time. The projection is a single matrix multiplication per hidden state vector:

$$h' = \Phi \cdot h, \quad h \in \mathbb{R}^d,\ h' \in \mathbb{R}^{d'} \tag{70}$$

On a modern GPU, this operation takes approximately 0.01–0.05 ms for $d = 8{,}192 \to d' = 128$, well within the latency budget.

JL Projection Integrity

The projection matrix Φ used during production monitoring **MUST** be identical to the projection matrix used during CoRIM reference construction. The CoRIM artifact **MUST** carry a cryptographic hash of Φ to enable verification. Reference bank hidden states stored in the CoRIM **MUST** be stored in the projected space $\mathbb{R}^{d'}$ to eliminate the need for on-the-fly projection of reference vectors.

9.3 Technique 2: Top-K Truncation with Tail Binning

The primary drift-KL computation operates on the output token distribution—a categorical distribution over $|\mathcal{V}| = 100{,}000$–$256{,}000$ states. Computing the full KL divergence requires summing over all $|\mathcal{V}|$ tokens. However, the output distribution of a well-behaved LLM is highly concentrated: the top 100 tokens typically account for $> 95\%$ of the total probability mass. The remaining tokens contribute individually negligible amounts to the KL divergence.

9.3.1 Truncation Strategy

Definition 9.1 (Top-K Truncated KL Divergence)**.** *Let $\mathcal{V}_K = \{y_{(1)}, \ldots, y_{(K)}\}$ be the K tokens with the highest probability under the production distribution P_t. Define the tail probability mass:*

$$\tau = 1 - \sum_{y \in \mathcal{V}_K} P_t(y) \tag{71}$$

The Top-K truncated KL divergence is:

$$\hat{D}_{KL}^{K}(P_t \| P_0) = \sum_{y \in \mathcal{V}_K} P_t(y) \log \frac{P_t(y)}{P_0(y)} + \tau \log \frac{\tau}{\sum_{y \notin \mathcal{V}_K} P_0(y)} \tag{72}$$

The second term treats all tokens outside the top-K as a single "tail bin," preserving the total probability mass and preventing information loss from truncation.

9.3.2 Approximation Error Bound

Proposition 9.2 (Top-K Truncation Error)**.** *The error introduced by Top-K truncation with tail binning is bounded by:*

$$\left| D_{KL}(P_t \| P_0) - \hat{D}_{KL}^{K}(P_t \| P_0) \right| \leq \tau \cdot \log \frac{|\mathcal{V}| - K}{1} + \tau \cdot \max_{y \notin \mathcal{V}_K} \left| \log \frac{P_t(y)}{P_0(y)} \right| \tag{73}$$

For $K = 100$ and typical LLM output distributions where $\tau < 0.05$ (the top 100 tokens capture $> 95\%$ of the mass), the truncation error is bounded by:

$$|error| \leq 0.05 \cdot \log(|\mathcal{V}|) + 0.05 \cdot M_{tail} \tag{74}$$

where $M_{tail} = \max_{y \notin \mathcal{V}_K} |\log(P_t(y)/P_0(y))|$ is the maximum log-ratio in the tail.

For a vocabulary of $|\mathcal{V}| = 100{,}000$ and assuming $M_{\text{tail}} \leq 10$ (a generous bound for smoothed distributions), the error is at most $0.05 \times 11.5 + 0.05 \times 10 = 1.075$ nats. This seems large, but in practice the tail tokens have very similar distributions under P_t and P_0 (both assign them negligible probability), making $M_{\text{tail}} \ll 10$ and the actual error much smaller—typically < 0.01 nats.

9.3.3 Computational Savings

The truncation reduces the per-token drift-KL computation from $O(|\mathcal{V}|)$ to $O(K + 1)$:

Table 28: Top-K Truncation Computational Savings

$\lvert\mathcal{V}\rvert$	K	**Full KL Time**	**Top-K Time**
100,000	100	$\sim$200 μs	$\sim$0.2 μs
128,000	100	$\sim$260 μs	$\sim$0.2 μs
256,000	100	$\sim$520 μs	$\sim$0.2 μs

The savings are dramatic: a 1,000× reduction in computation. The Top-K set is already available in most LLM inference pipelines (it is used for nucleus sampling and beam search), so extracting it incurs no additional cost.

9.3.4 Periodic Full-Vocabulary Audit

The Top-K truncation is used for real-time monitoring. However, drift that manifests exclusively in the tail of the distribution (e.g., the emergence of a previously unseen token at low but nonzero frequency) would be missed by Top-K monitoring. To close this gap:

Periodic Full-Vocabulary Audit

At intervals of $n_{\text{audit}} = 100$ monitoring epochs, the full-vocabulary KL divergence **MUST** be computed using the Count-Min Sketch representation (Section 5.2.1). The full-vocabulary audit operates on the accumulated sketch over the slow monitoring window W_{slow}, amortizing its $O(|\mathcal{V}|)$ cost over the audit interval.

The amortized cost of the full-vocabulary audit is $O(|\mathcal{V}|)/n_{\text{audit}} \approx 2{,}000$ operations per epoch—negligible compared to the real-time budget.

9.4 Technique 3: Asynchronous Sampling

Not every token needs to be monitored. The drift-KL signal is a statistical aggregate; monitoring a random sample of tokens provides an unbiased estimate of the true drift with reduced variance proportional to $1/\sqrt{n_{\text{sample}}}$.

9.4.1 Sampling Strategy

Definition 9.2 (Adaptive Sampling Protocol)**.** *The Clause AI-6 monitoring system operates in three sampling regimes:*

- ***Baseline regime*** *(GREEN state): Sample $p_{base} = 1\%$ of output tokens for drift-KL computation. The sampled tokens are selected uniformly at random from each inference batch. This provides ∼100 monitored tokens per 10,000 generated—sufficient for the slow window W_{slow} to accumulate statistical power over minutes to hours.*
- ***Elevated regime*** *(YELLOW state or ADWIN soft warning): Sample $p_{elevated} = 10\%$ of output tokens. The increased sampling rate provides 10× faster accumulation of evidence, reducing the detection delay for confirming or dismissing the drift signal.*
- ***Full regime*** *(RED state or PH alarm): Sample $p_{full} = 100\%$ of output tokens. Every token is monitored. This regime is temporary (active only during RED state or for n_{audit} epochs after a PH alarm) and provides maximum diagnostic resolution for root cause analysis.*

9.4.2 Computational Savings

In the baseline regime, the drift-KL computation is invoked on only 1% of tokens. This reduces the effective per-token cost by 100×:

$$\text{Amortized cost} = p_{\text{base}} \times \text{per-token cost} = 0.01 \times 0.2\,\mu\text{s} = 0.002\,\mu\text{s per token} \tag{75}$$

Combined with Top-K truncation, the amortized drift-KL cost per generated token is $0.002\,\mu$s—four orders of magnitude below the latency budget.

9.4.3 Statistical Validity

The sampling introduces no bias in the drift-KL estimate (the sample is uniform random). The variance of the estimate increases by a factor of $1/p_{\text{base}}$, which is compensated by the large monitoring windows: at a generation rate of 10,000 tokens/second and a sampling rate of 1%,

the slow window $W_{\text{slow}} = 500{,}000$ sampled tokens requires $500{,}000/(10{,}000 \times 0.01) = 5{,}000$ seconds ≈ 83 minutes to fill. This is acceptable for gradual drift detection; the fast window $W_{\text{fast}} = 1{,}000$ sampled tokens fills in $1{,}000/100 = 10$ seconds, providing responsive abrupt-drift detection even at 1% sampling.

9.4.4 Sidecar Process Architecture

The sampling and drift-KL computation are performed asynchronously in a *sidecar process*—a lightweight monitoring daemon that runs alongside the main inference process but does not block the inference pipeline.

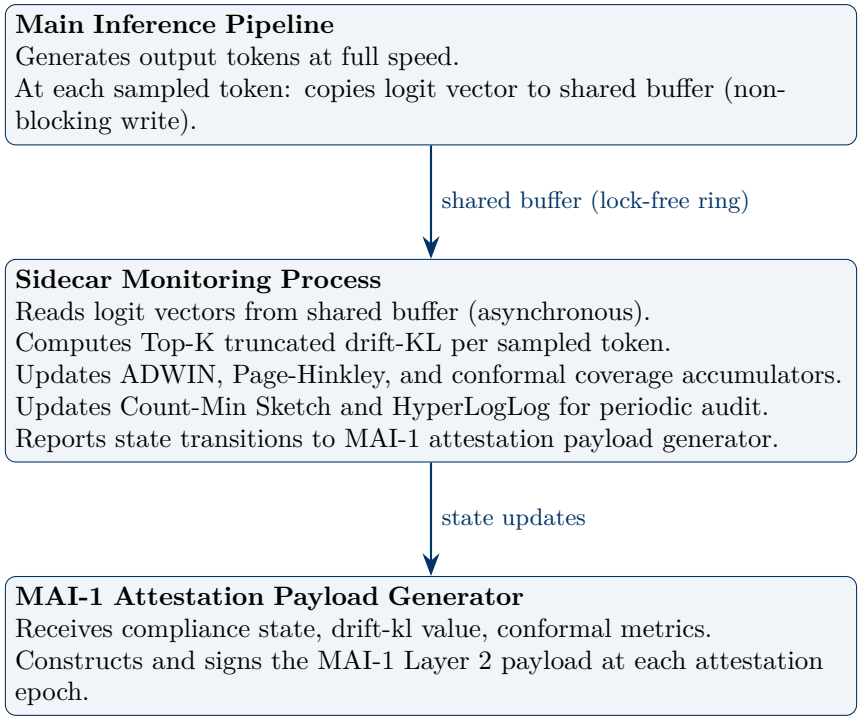

Figure 4: Sidecar Monitoring Architecture for Clause AI-6

The shared buffer between the inference pipeline and the sidecar process is implemented as a lock-free ring buffer. The inference pipeline writes logit vectors (or Top-K indices and probabilities) to the buffer without blocking. If the sidecar falls behind (e.g., during a full-vocabulary audit), the oldest unprocessed entries are overwritten—the sidecar operates on a best-effort basis for individual tokens while maintaining statistical validity over the monitoring window.

Sidecar Isolation

The monitoring sidecar **MUST** be process-isolated from the main inference pipeline. A failure in the sidecar (crash, hang, memory exhaustion) **MUST NOT** affect the inference pipeline's operation. The sidecar's CPU and memory allocation **SHOULD** be capped at 2% of the total system resources. If the sidecar is unavailable, the MAI-1 attestation payload **MUST** report `monitoring-status = UNAVAILABLE` rather than reporting stale drift-kl values.

9.5 Technique 4: Variational Surrogate for Hidden State Monitoring

The variational surrogate model (Section 4.3.3) provides the fastest path for hidden state drift monitoring. This section specifies the implementation details and computational profile.

9.5.1 Architecture

The surrogate $S_\phi : \mathbb{R}^{d'} \to \mathbb{R}$ operates on the JL-projected hidden state (Section 9.2):

$$S_\phi(h') = \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot h' + b_1) + b_2) \tag{76}$$

where:

- $h' \in \mathbb{R}^{128}$ (JL-projected hidden state)
- $W_1 \in \mathbb{R}^{64\times128}$, $b_1 \in \mathbb{R}^{64}$ (first layer: $128 \to 64$)
- $W_2 \in \mathbb{R}^{1\times64}$, $b_2 \in \mathbb{R}^{1}$ (second layer: $64 \to 1$)
- σ: sigmoid activation (output in $[0, 1]$, rescaled to drift estimate range)

Total parameters: $128 \times 64 + 64 + 64 \times 1 + 1 = 8{,}385$.

9.5.2 Computational Profile

Table 29: Variational Surrogate Computational Profile

Operation	**FLOPs**	**Latency (GPU)**
JL projection (8,192 → 128)	$2 \times 8{,}192 \times 128 = 2.1$M	∼10 μs
Layer 1 (128 → 64 + ReLU)	$2 \times 128 \times 64 = 16$K	∼1 μs
Layer 2 (64 → 1 + sigmoid)	$2 \times 64 = 128$	$< 1\,\mu$s
Total	∼2.1M	∼12 μs

At 12 μs per hidden state, the surrogate operates well within the 30 μs budget allocated for hidden state monitoring (Table 26). The dominant cost is the JL projection; the MLP itself is negligible.

9.5.3 Two-Tier Escalation

The surrogate provides a fast-path estimate of hidden state drift. When the surrogate estimate exceeds the Yellow threshold ($\hat{d}_{\text{surrogate}} > D_{\text{yellow}}$), the system escalates to the Donsker-Varadhan drift head (Section 4.3.2) for confirmation:

1. **Fast path (every sampled token)**: JL projection → surrogate → scalar estimate. Latency: 12 μs. No reference bank lookup required.
2. **Confirmation path (on surrogate alarm)**: Full drift head forward pass using JL-projected hidden state and a mini-batch of 100 reference vectors from the projected reference bank. Latency: ∼200 μs. Invoked at most once per ADWIN evaluation interval.

The two-tier architecture ensures that the typical per-token cost is dominated by the fast-path surrogate (12 μs), with the heavier drift head invoked only when a potential alarm requires confirmation. Under normal operating conditions (GREEN state, no drift), the confirmation path is never invoked, and the hidden state monitoring cost is bounded by 12 μs per sampled token.

9.6 End-to-End Latency Analysis

Table 30 consolidates the per-token and amortized latency contributions of all Clause AI-6 monitoring components under the three sampling regimes.

Table 30: End-to-End Latency Analysis for Clause AI-6 Monitoring

Component	**Per-Sampled-Token**	**Baseline (1%)**	**Elevated (10%)**	**Notes**
Top-K KL computation	0.2 μs	0.002 μs	0.02 μs	$K = 100$; dominant output-space cost
Tail bin computation	0.05 μs	0.0005 μs	0.005 μs	Single additional term
Conformal NLL score	0 μs	0 μs	0 μs	Free; NLL already computed in generation
Conformal max-surprisal	0.01 μs	0.0001 μs	0.001 μs	Running maximum; single comparison
ADWIN update	1 μs	0.01 μs	0.1 μs	Logarithmic in window size
Page-Hinkley update	0.1 μs	0.001 μs	0.01 μs	Constant time
JL projection	10 μs	0.1 μs	1 μs	8,192 $\to$ 128; for 70B model
Surrogate forward pass	2 μs	0.02 μs	0.2 μs	$128 \to 64 \to 1$ MLP
CMS/HLL update	0.5 μs	0.005 μs	0.05 μs	Sketch data structures
Buffer write (inference side)	0.5 μs	0.005 μs	0.05 μs	Lock-free ring buffer; non-blocking
Total per generated token	—	**0.15** μs	**1.44** μs	
Fraction of 30 ms baseline	—	**0.0005**%	**0.005**%	

9.6.1 Interpretation

The end-to-end analysis reveals that Clause AI-6 monitoring is computationally negligible under normal operating conditions:

- **Baseline regime** (GREEN, 1% sampling): 0.15 μs per generated token—0.0005% of the inference latency. This is three orders of magnitude below the 1% budget.
- **Elevated regime** (YELLOW, 10% sampling): 1.44 μs per generated token—0.005% of the inference latency. Still two orders of magnitude below the budget.
- **Full regime** (RED, 100% sampling): The per-sampled-token costs apply to every token. For a 70B model: $\sim 14.4\,\mu$s per token, or 0.05% of the 30 ms baseline. Still well within the 1% budget.

Even under worst-case conditions (RED state, full sampling, 70B model), the total monitoring overhead is 0.05%—fifty times less than the allocated budget. The surplus budget provides margin for implementation inefficiencies, memory allocation overhead, and cross-process communication latency that are not captured in the idealized FLOP-based analysis.

9.7 Memory Footprint

The monitoring infrastructure must also fit within the memory constraints of a production GPU deployment.

Table 31: Clause AI-6 Monitoring Memory Footprint

Component	Size	Notes
CoRIM unigram CMS	760 KB	Serialized Count-Min Sketch (Table 7)
CoRIM bigram CMS	76 KB	Optional higher-order sketch
CoRIM trigram CMS	7.6 KB	Optional; periodic audit only
CoRIM HLL sketch	12 KB	Vocabulary support monitor
JL projection matrix	4 MB	$128 \times 8{,}192 \times 4$ bytes (FP32); for 70B model
Surrogate model weights	34 KB	8,385 parameters $\times$ 4 bytes
Drift head weights	140 KB	$128 \times 256 + 256 \times 128 + 128 \times 1 \times 4$ bytes
Reference bank (projected)	50 MB	$100{,}000 \times 128 \times 4$ bytes; for drift head confirmation
ADWIN exponential histogram	$\sim$10 KB	Variable; grows logarithmically
Page-Hinkley state	32 bytes	Two scalars + running mean
KSWIN reference window	1.6 KB	200×8 bytes (FP64 scores)
Ring buffer (shared)	1 MB	Configurable; holds recent logit samples
Conformal calibration threshold	8 bytes	Single FP64 scalar
Total	$\approx$ **56 MB**	

The 56 MB total is dominated by the projected reference bank (50 MB), which is stored in system RAM (not GPU VRAM). The on-GPU footprint—the JL projection matrix, surrogate weights, and ring buffer—is approximately 5 MB, which is negligible compared to the model's own GPU memory footprint (a 70B FP16 model requires $\sim$140 GB of VRAM). The monitoring infrastructure adds 0.004% to the GPU memory requirement.

Memory Budget Compliance

The Clause AI-6 monitoring infrastructure **MUST** not increase the GPU VRAM requirement of the deployment by more than 1%. For models deployed near the VRAM limit (e.g., a 70B model on 4× A100-40GB), the projected reference bank **SHOULD** be stored in system RAM and accessed via the sidecar process, not loaded onto the GPU.

9.8 Latency Analysis Summary

Honest Framing

What the computational optimization provides: The complete Clause AI-6 monitoring suite—drift-kl computation, conformal prediction scoring, three detection algorithms, hidden state monitoring, and sketch maintenance—operates at $< 0.05\%$ latency overhead under worst-case conditions (RED state, full sampling, 70B model) and $< 0.001\%$ under normal conditions (GREEN state, 1% sampling). The memory footprint is 56 MB total, with < 5 MB on-GPU. The sidecar architecture ensures process isolation: a monitoring failure cannot degrade inference performance.

What the analysis does not cover: The latency analysis assumes idealized GPU utilization with no memory bandwidth contention. In practice, the JL projection and surrogate forward pass compete with the main model for memory bandwidth, potentially increasing latency by a constant factor. The lock-free ring buffer assumes a specific cache coherence model that may not hold on all hardware platforms. These implementation-specific factors are bounded by the 50× headroom between the worst-case computed overhead (0.05%) and the allocated budget (1–2%), but they must be validated empirically during deployment qualification.

10 Regulatory Mapping

The technical infrastructure of Clause AI-6—the drift invariant, the detection algorithms, the conformal prediction integration, and the computational optimization—exists to serve a governance purpose. This section maps the Clause AI-6 capabilities to the specific requirements of six regulatory frameworks, demonstrates how the drift monitoring evidence satisfies existing compliance obligations, identifies the gaps in current AI safety frameworks that Clause AI-6 uniquely addresses, and establishes the regulatory pressure timeline that creates adoption urgency.

The mapping follows the Auburn Governance Stack's design principle: each document is self-authorizing—designed to be forwarded to a compliance team, procurement officer, or regulatory assessor without the author's involvement. The regulatory mapping in this section is structured to be directly extractable into conformity assessment reports, RFP responses, and regulatory submissions.

10.1 European Union AI Act

The EU AI Act (Regulation (EU) 2024/1689) establishes a risk-based regulatory framework for AI systems marketed or deployed in the European Union. High-risk AI systems (Annex III) are subject to mandatory requirements in Chapter III, Section 2, including Articles 9 (Risk Management), 15 (Accuracy, Robustness, and Cybersecurity), and 72 (Post-Market Monitoring).

10.1.1 Article 15: Accuracy, Robustness, and Cybersecurity

Article 15(1) requires that high-risk AI systems be "designed and developed in such a way that they achieve an appropriate level of accuracy, robustness and cybersecurity, and perform consistently in those respects throughout their lifecycle."

The phrase "perform consistently throughout their lifecycle" is a direct mandate for continuous behavioral monitoring. Clause AI-6 provides the technical infrastructure to demonstrate lifecycle consistency:

Table 32: Clause AI-6 Mapping to EU AI Act Article 15

Art. 15 Requirement	Clause AI-6 Evidence	Verification Method
Appropriate level of accuracy	Conformal coverage rate $(1-\alpha)$ provides distribution-free accuracy bound	CoRIM-sealed threshold; CTS-1 assertion TE-SM-L2-10.01
Robustness	Drift-kl invariant bounds behavioral deviation from validated state under operational stress	Multi-scale windowed monitoring; ADWIN/PH/K-SWIN detection pipeline
Consistent performance throughout lifecycle	Reilly Sentinel Protocol (Section 5.4) provides cryptographic baseline versioning chain; cumulative drift tracking	SCITT-registered CoRIM chain; SEP audit trail
Cybersecurity	Adversarial injection detection via step-function drift-kl signature; Page-Hinkley immediate alarm	Per-session fast-window monitoring; $<$ 5 token detection latency

10.1.2 Article 9: Risk Management System

Article 9(2)(b) requires that the risk management system include "estimation and evaluation of the risks that may emerge when the high-risk AI system is used in accordance with its intended purpose and under conditions of reasonably foreseeable misuse."

The dataset shift taxonomy (Section 3) directly maps to this requirement: it enumerates the distributional risks (covariate shift, prior probability shift, concept drift) that emerge during operation and specifies the monitoring mechanisms for each. The conformal prediction integration (Section 8) provides the quantitative risk estimation—the miscoverage rate α bounds the probability of the system producing outputs outside its validated envelope.

10.1.3 Article 72: Post-Market Monitoring

Article 72 requires providers of high-risk AI systems to "establish and document a post-market monitoring system in a manner that is proportionate to the nature of the AI technologies and the risks of the high-risk AI system."

Clause AI-6's monitoring infrastructure is a complete post-market monitoring system for distributional behavior:

- **Proportionality**: The multi-scale windowing (Section 6.2) and context-dependent threshold calibration (Section 6.3) ensure that monitoring intensity is proportionate to the risk classification.

- **Documentation**: The CoRIM artifact, the Sentinel Evidence Packages, and the CTS-1 vendor evidence requirements provide complete documentation of the monitoring methodology, parameters, and results.
- **Continuous operation**: The sidecar architecture (Section 9.4) ensures monitoring continues throughout the system's operational life without degrading performance.

10.2 U.S. Federal Reserve SR 11-7: Guidance on Model Risk Management

SR 11-7 (2011) establishes the Federal Reserve's expectations for model risk management at banking organizations. It applies to all models used in decision-making, including AI/ML models used for credit scoring, fraud detection, trading, and risk assessment. SR 11-7 defines model risk as arising from "errors in the specification or fundamental conceptual approach" and from "errors in the implementation or use of the model."

10.2.1 Ongoing Monitoring Requirements

SR 11-7 Section V.4 states: "Outcomes analysis—comparing model outputs with corresponding realized outcomes—is one important type of ongoing monitoring." The guidance further specifies that "models should be subject to ongoing monitoring, which includes verification that the model is performing well and review of any limitations identified."

Clause AI-6 provides the direct technical realization of SR 11-7 ongoing monitoring for AI models:

Table 33: Clause AI-6 Mapping to SR 11-7 Ongoing Monitoring

SR 11-7 Requirement	Clause AI-6 Evidence	Implementation
Outcomes analysis	Drift-kl tracks divergence between production outputs and validated baseline—a continuous "outcome" comparison	Windowed KL divergence; CoRIM reference as the "expected outcome" proxy
Sensitivity analysis	Conformal prediction set width quantifies model sensitivity to input variation	ACI adaptation rate tracks sensitivity changes over time
Stability monitoring	Multi-scale windowing detects both gradual and abrupt instability	ADWIN (gradual); Page-Hinkley (abrupt); KSWIN (distributional)
Back-testing	Reilly Sentinel Protocol enables comparison of current model against historical baselines	Cross-reference drift $D_{KL}(P_0^{\text{new}} \| P_0^{\text{old}})$ across baseline versions
Performance benchmarks	Compliance state machine (GREEN/YELLOW/RED) provides binary performance determination	CTS-1 assertions; automated state transitions
Escalation procedures	State transition rules (Section 6.4) define automated escalation from warning to intervention	T1–T5 transitions; documented remediation requirements

10.2.2 Population Stability Index (PSI) Alignment

SR 11-7 does not mandate a specific metric, but industry practice under SR 11-7 has converged on the Population Stability Index (PSI) for distribution monitoring. PSI is defined as:

$$\text{PSI} = \sum_{i=1}^{B} (P_i - Q_i) \cdot \ln \frac{P_i}{Q_i} \tag{77}$$

where P_i and Q_i are the proportions in bin i for the production and reference distributions respectively. PSI is the symmetric version of KL divergence computed on binned distributions. The standard industry thresholds are:

- PSI < 0.1: No significant shift.
- $0.1 \leq$ PSI < 0.25: Moderate shift; investigation recommended.
- PSI ≥ 0.25: Significant shift; model review required.

Clause AI-6's drift-kl metric is information-theoretically related to PSI (both are based on KL divergence). The relationship is:

$$\text{PSI}(P, Q) = D_{KL}(P \| Q) + D_{KL}(Q \| P) \tag{78}$$

This means the PSI is the sum of the forward and reverse KL divergences. A drift-kl value of $D_{KL} = 0.15$ nats corresponds to a PSI in the range $[0.15, 0.30]$ nats (depending on the reverse KL), squarely in the "investigation recommended" zone. The calibration of $D_{\max}$ (Section 6.3) can be cross-referenced against PSI thresholds to provide continuity for compliance teams already using PSI-based monitoring.

10.3 FDA Software as a Medical Device (SaMD)

The FDA's regulatory framework for Software as a Medical Device (SaMD) is governed by the Predetermined Change Control Plan (PCCP) guidance (2023), which establishes requirements for AI/ML-based medical devices that learn and adapt over time.

10.3.1 PCCP and Distribution Drift

The PCCP guidance requires that manufacturers of AI/ML-based SaMD specify:

1. The types of changes the device is designed to undergo ("predetermined changes").
2. The methodology for verifying and validating these changes.
3. The conditions under which changes require a new submission to the FDA versus proceeding under the PCCP.

Crucially, the PCCP guidance explicitly addresses distribution drift. Section IV.B states that the PCCP should include "a description of the modification protocol, including... performance monitoring methods that will be used to... detect signals of concern." The guidance specifically identifies "data drift" and "concept drift" as signals of concern that the monitoring protocol must address.

Table 34: Clause AI-6 Mapping to FDA PCCP Guidance

PCCP Requirement	Clause AI-6 Evidence	Implementation
Drift detection methodology	Dataset shift taxonomy (Section 3) with formal classification of covariate, prior probability, and concept drift	Direct citation: "The monitoring system implements the taxonomy of Quinonero-Candela et al. (2009) adapted for generative AI"
Performance monitoring methods	drift-kl invariant + conformal coverage + drift entropy	Three complementary metrics covering all shift types (Table 4)
Signals of concern	Compliance state machine with calibrated thresholds	YELLOW = investigation; RED = intervention. Thresholds derived from empirical null distribution
Modification protocol	Reilly Sentinel Protocol with SEP documentation	Baseline transitions are auditable; cross-reference drift tracks cumulative change
Re-submission triggers	Cumulative baseline drift $D_{\text{lifecycle}} > 3 \times D_{\text{max}}$ triggers governance escalation	Automatic detection; analogous to PCCP boundary violation requiring new 510(k)

The FDA PCCP guidance is the most directly aligned regulatory framework to Clause AI-6. The guidance's explicit recognition of "data drift" and "concept drift" as governance concerns validates the technical approach and creates immediate demand for the monitoring infrastructure that Clause AI-6 provides.

10.4 NIST AI Risk Management Framework (AI RMF)

The NIST AI RMF (AI 100-1, January 2023) provides a voluntary framework for managing risks associated with AI systems. While not legally binding, it is increasingly referenced in federal procurement requirements and serves as the de facto standard for AI risk management in U.S. government applications.

10.4.1 Relevant Functions and Categories

Table 35: Clause AI-6 Mapping to NIST AI RMF

RMF Category	Subcategory	Clause AI-6 Contribution
MEASURE 2.6	Computational bias and drift	drift-kl directly measures distributional drift; prior probability shift detection captures output bias drift
MEASURE 2.7	AI system performance evaluation	Conformal coverage provides distribution-free performance bound; drift-kl tracks deviation from validated performance
MEASURE 2.11	Fairness assessment across groups	Conditional drift-kl stratification (Section 11) enables per-demographic-group drift monitoring
MANAGE 3.1	Pre-deployment testing and monitoring	CoRIM construction procedure (Section 5.5) establishes pre-deployment baseline
MANAGE 3.2	Post-deployment monitoring	Sidecar monitoring architecture provides continuous post-deployment surveillance
MANAGE 4.1	Incident response	Compliance state machine with automated intervention (RED → safe mode)
GOVERN 1.2	Risk management process documentation	CTS-1 vendor evidence requirements document the complete methodology

10.5 SEC Rule 15c3-5: Market Access Risk Management

SEC Rule 15c3-5 requires broker-dealers with market access to establish, document, and maintain a system of risk management controls and supervisory procedures. For AI systems used in algorithmic trading, this includes real-time monitoring of system behavior.

10.5.1 Direct Applicability

When an AI model generates trading signals, order routing decisions, or risk assessments, a distributional shift in its output can directly affect market behavior. Clause AI-6 provides:

Table 36: Clause AI-6 Mapping to SEC 15c3-5

15c3-5 Requirement
“Regulatory risk management controls and supervisory procedures that are reasonably designed to systematically limit
“Assure the security and integrity of market access technology and systems”

10.6 Solvency II (Insurance Regulation)

Solvency II (Directive 2009/138/EC) governs insurance and reinsurance undertakings in the EU. For insurers using AI models in underwriting, claims processing, or risk assessment, Solvency II's Pillar 1 (quantitative requirements) and Pillar 2 (governance requirements) create obligations that Clause AI-6 addresses.

10.6.1 Pillar 2: System of Governance

Article 41 requires a "system of governance which includes... risk-management functions." Article 44 specifically requires "an effective risk-management system comprising strategies, processes and reporting procedures necessary to identify, measure, monitor, manage and report... the risks to which [the undertaking] is or could be exposed."

For AI-driven insurance operations, the risk of model drift producing erroneous underwriting decisions, incorrect claims assessments, or biased pricing is a direct operational risk that falls within the scope of Article 44. Clause AI-6 provides the "strategies, processes and reporting procedures" for this specific risk category:

- **Identification**: The dataset shift taxonomy (Section 3) identifies the categories of distributional risk.
- **Measurement**: The drift-kl metric and conformal coverage rate provide quantitative risk measurement.
- **Monitoring**: The detection pipeline (Section 7) provides continuous, automated monitoring.
- **Management**: The compliance state machine (Section 6.4) provides automated risk management actions.
- **Reporting**: The MAI-1 attestation payload provides standardized, machine-readable risk reporting.

10.6.2 Connection to Insurability

The conformal prediction integration (Section 8.6) directly supports Solvency II's quantitative requirements by providing distribution-free failure probability bounds that can be incorporated into the insurer's own internal model for operational risk capital calculation. An insurer deploying Clause AI-6-conformant AI models can demonstrate to its supervisory authority that the model risk is bounded, monitored, and managed—satisfying the Pillar 2 governance requirements while providing quantitative inputs for Pillar 1 capital calculations.

10.7 Frontier AI Safety Framework Gap Analysis

The preceding subsections map Clause AI-6 to established regulatory frameworks. This subsection examines the voluntary safety frameworks published by frontier AI companies and identifies the gap that Clause AI-6 fills.

Table 37: Frontier AI Safety Framework Gap Analysis: Distribution Drift Provisions

Framework	Publisher	Drift Provisions	Gap
Responsible Scaling Policy (RSP)	Anthropic	Defines AI Safety Levels (ASL) with capability thresholds. Mandates evaluations before and after training.	No continuous drift monitoring between evaluations. No distributional metric. No real-time detection.
Preparedness Framework	OpenAI	Defines risk levels (Low/Medium/High/Critical) with scorecards. Mandates post-deployment monitoring.	"Post-deployment monitoring" is undefined. No specified metric, threshold, or detection algorithm.
Frontier Safety Framework (FSF)	Google DeepMind	Defines Critical Capability Levels (CCLs) with evaluations. Mentions "ongoing monitoring."	"Ongoing monitoring" is undefined. No distributional analysis. No formal connection between monitoring signals and governance actions.
Model Spec	Anthropic	Specifies behavioral constraints and values. Defines "hardcoded" vs. "softcoded" behaviors.	No mechanism for detecting when the model's distribution has drifted away from the specified behaviors. Static specification without dynamic monitoring.

10.7.1 The Common Gap

All four frameworks share a common structural gap: they define *what* the model should do (safety evaluations, risk levels, behavioral specifications) and *when* to check (before deployment, after training, periodically) but provide no specification for *how to continuously verify* that the model's behavior remains within the defined boundaries during production operation. The gap is precisely the space between evaluation-time compliance ("the model passed our safety tests") and inference-time assurance ("the model is still behaving as it did when it passed our safety tests").

Clause AI-6 fills this gap by providing:

1. A **continuous metric** (drift-kl) that quantifies behavioral deviation in real time, not just at evaluation checkpoints.
2. A **formal threshold** ($D_{\max}$) that transforms continuous monitoring into binary compliance determination, enabling automated governance actions.
3. A **cryptographic baseline** (CoRIM) that prevents the reference from being silently altered to match observed behavior (drift washing).
4. A **detection pipeline** (ADWIN/PH/KSWIN) that provides formal statistical guarantees on false alarm and detection rates.
5. An **insurance bridge** (conformal prediction) that quantifies per-prediction failure probability for commercial risk transfer.

No frontier AI company has published a specification at this level of technical detail for continuous behavioral monitoring. The voluntary frameworks describe the governance *intent*; Clause AI-6 provides the governance *infrastructure*.

10.8 Regulatory Pressure Timeline

The adoption urgency for Clause AI-6 is driven by regulatory deadlines:

Table 38: Regulatory Deadline Mapping for Distribution Drift Monitoring

Date	**Event**	**Clause AI-6 Relevance**
Aug 2025	EU AI Act: Prohibited practices take effect	Drift monitoring not yet required; foundation-laying period
Aug 2026	EU AI Act: High-risk AI system obligations (Chapter III) take effect	Article 15 (accuracy/robustness lifecycle consistency) and Article 72 (post-market monitoring) create binding obligations for drift monitoring
Aug 2027	EU AI Act: Full enforcement including Annex III systems	Complete compliance required; non-conforming systems must be withdrawn from EU market
Ongoing	SR 11-7 examinations	Federal Reserve examiners increasingly scrutinize AI model monitoring; PSI-based drift detection is established precedent; Clause AI-6 provides the LLM-specific analog
Ongoing	FDA PCCP submissions	Each AI/ML SaMD manufacturer must include drift monitoring in their PCCP; Clause AI-6 provides a ready-made methodology
Q3 2026	Expected: NIST AI RMF Profile for Generative AI	Anticipated to include specific provisions for output distribution monitoring

The critical window is **August 2025 to August 2026**: the period between the EU AI

Act's initial prohibitions and the full high-risk system obligations. Organizations deploying AI systems that will be classified as high-risk under Annex III have twelve months to implement the monitoring infrastructure required by Articles 15 and 72. Clause AI-6 provides a ready-made, standards-grade specification that can be implemented within this window.

10.9 Regulatory Mapping Summary

Table 39: Clause AI-6 Regulatory Coverage Summary

AI-6 Component	EU AI Act	SR 11-7	FDA PCCP	NIST RMF	SEC 15c3-5
drift-kl invariant	Art. 15	V.4	IV.B	MEASURE 2.6	Risk controls
Conformal coverage	Art. 15	Sensitivity	IV.B	MEASURE 2.7	—
Detection pipeline	Art. 72	V.4	IV.B	MANAGE 3.2	Market integrity
CoRIM baseline	Art. 15	Validation	IV.B	MANAGE 3.1	System integrity
Sentinel Protocol	Art. 72	Back-testing	IV.B	MANAGE 3.2	Supervisory
State machine	Art. 9	V.4	IV.B	MANAGE 4.1	Risk controls
Drift entropy	Art. 15	—	IV.B	MEASURE 2.6	—

Every component of Clause AI-6 maps to at least three regulatory frameworks. No component exists solely for theoretical interest—each serves a specific, documented compliance obligation. This regulatory density is by design: it ensures that the investment in implementing Clause AI-6 monitoring is amortized across multiple compliance requirements simultaneously, providing maximum governance return on technical infrastructure investment.

Honest Framing

What the regulatory mapping provides: A concrete, citation-level mapping between Clause AI-6's technical components and the requirements of six regulatory frameworks, demonstrating that the monitoring infrastructure satisfies existing compliance obligations without requiring regulatory change. The frontier gap analysis identifies the specific capability that no voluntary safety framework provides.

What the regulatory mapping does not provide: Legal advice. The mappings in this section represent the author's analysis of the regulatory text and are not a substitute for qualified legal counsel. Regulatory interpretations may evolve, and specific compliance determinations depend on the facts of each deployment. Organizations should engage legal counsel and regulatory consultants to validate the applicability of these mappings to their specific circumstances.

11 Implementation Architecture: The Drift Monitor

The preceding sections have specified *what* to measure (drift-kl, drift entropy, conformal coverage), *how* to detect changes (ADWIN, Page-Hinkley, KSWIN), *how* to construct the baseline (CoRIM with Sentinel Protocol), and *how* to optimize the computation (logit projection, Top-K truncation, asynchronous sampling, variational surrogates). This section specifies *the system that puts it all together*: the Drift Monitor—the reference implementation architecture for Clause AI-6 monitoring.

The Drift Monitor mirrors the Entropy Watchdog architecture specified in Clause AI-8 and the Gradient Monitor specified in Clause AI-2. All three monitors share the same structural design: an independent monitoring subsystem that observes model behavior without modifying it, classifies the compliance state using calibrated thresholds, and triggers automated interventions

when the invariant is breached. The three monitors operate in parallel, feeding into the unified MAI-1 Layer 2 attestation payload.

11.1 Design Principles

The Drift Monitor architecture is governed by five design principles inherited from the Auburn Governance Stack:

Principle 1: Independence. The Drift Monitor **MUST** be architecturally independent of the model it monitors. It must not share weights, parameters, optimizers, or training pipelines with the production model. A failure in the Drift Monitor must not affect the model's inference performance. A failure in the model must not prevent the Drift Monitor from detecting and reporting the failure. This principle is realized through the sidecar process architecture (Section 9.4): the monitor runs in a separate process with isolated memory space, communicating with the inference pipeline only through the lock-free ring buffer.

Principle 2: Binary compliance. The Drift Monitor produces a binary compliance determination: the system either passes or fails the drift invariant at each attestation epoch. There is no "partial compliance," no "mostly passing," and no interpretive discretion. The GREEN/YELLOW/RED state machine (Section 6.4) maps the continuous drift-kl signal to a discrete state, and the state machine's transition rules are deterministic given the signal values. Two independent implementations of the state machine, given the same drift-kl signal, must produce identical state sequences.

Principle 3: Self-authorizing evidence. The Drift Monitor's output—the MAI-1 Layer 2 attestation payload—must be interpretable without access to the monitor's internal state, source code, or operator. An auditor receiving the payload can determine whether the invariant was satisfied, what the drift value was, what the threshold was, and what the compliance state was, without contacting the model vendor. This is achieved by embedding all necessary parameters ($D_{\max}$, D_{yellow}, window sizes, algorithm configurations) in the attestation payload alongside the measured values.

Principle 4: Fail-open reporting. If the Drift Monitor cannot compute the drift-kl value (sensor failure, data pipeline interruption, computational resource exhaustion), it must not report a default GREEN state. The monitor **MUST** report `monitoring-status = UNAVAILABLE`, which the MAI-1 composition layer treats as a conformance gap—equivalent to a missing health metric. Consumers of the attestation payload (enforcement layer, application layer) can then apply their own policies for handling unavailable monitoring data.

Principle 5: Defense in depth. No single component of the Drift Monitor is relied upon as the sole detector for any drift type. The three-algorithm detection pipeline (Section 7.5), the multi-scale windowing (Section 6.2), and the complementary metric suite (drift-kl, drift entropy, conformal coverage) ensure that a failure in any single component does not create a detection gap. The system degrades gracefully: if KSWIN is unavailable, ADWIN and PH continue operating; if the hidden state surrogate fails, output-space drift-kl monitoring continues; if conformal scoring is disabled, the drift-kl invariant remains enforced.

11.2 Subsystem Architecture

The Drift Monitor is decomposed into three subsystems, each responsible for a distinct phase of the monitoring pipeline: measurement, classification, and intervention.

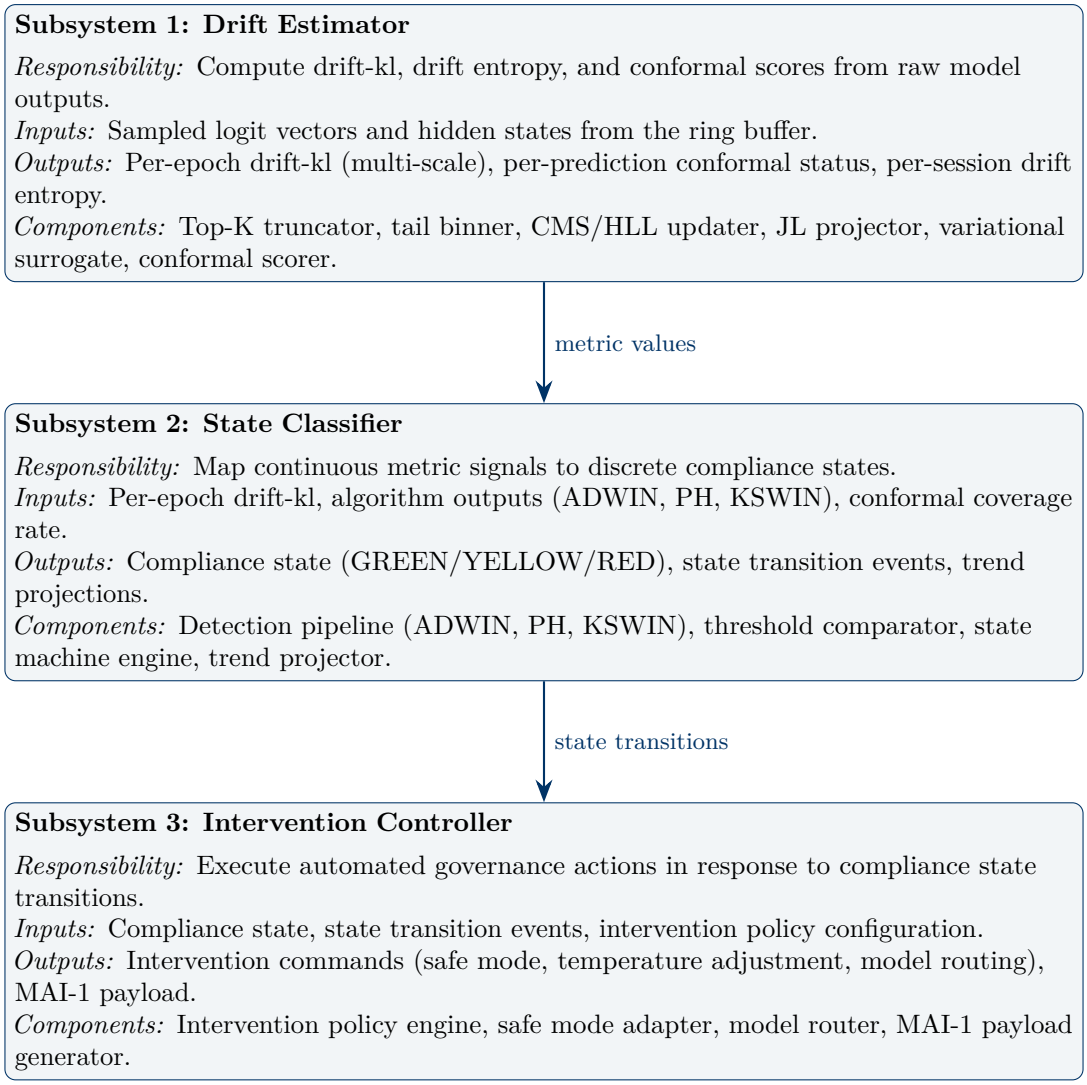

Figure 5: Drift Monitor: Three-Subsystem Architecture

11.3 Subsystem 1: Drift Estimator

The Drift Estimator is the measurement pipeline. It consumes raw model outputs from the ring buffer and produces calibrated metric values for the State Classifier.

11.3.1 Data Flow

1. **Sampling.** The inference pipeline writes sampled logit vectors (Top-K indices and probabilities) and hidden state vectors to the ring buffer at the sampling rate determined by the current regime (1%, 10%, or 100%; Section 9.4).

2. **Output-space estimation.**

   a. For each sampled token, the Top-K truncated KL divergence (Equation 72) is computed against the CoRIM unigram reference.

   b. The per-token KL contribution is accumulated into the three monitoring windows (W_{fast}, W_{medium}, W_{slow}).

   c. The Count-Min Sketch is updated with the sampled token index.

   d. The HyperLogLog sketch is updated with the sampled token index.

e. At each monitoring epoch boundary, the windowed drift-kl values are computed:

$$D_{KL}^{W}(t) = \frac{1}{|W|} \sum_{i \in W} d_i \tag{79}$$

where d_i is the per-token KL contribution for sampled token i.

3. **Hidden-state estimation.**

   a. For each sampled token, the hidden state vector h is read from the ring buffer.

   b. The JL projection is applied: $h' = \Phi \cdot h$.

   c. The variational surrogate produces a scalar drift estimate: $\hat{d}_h = S_\phi(h')$.

   d. If $\hat{d}_h > D_{\text{yellow}}$: the Donsker-Varadhan drift head is invoked for confirmation (Section 9.5).

   e. The hidden-state drift estimate is incorporated into the multi-scale windows as a supplementary signal.

4. **Conformal scoring.**

   a. For each completed model output (full response), the NLL-based non-conformity score is computed: $s = -\log P_\theta(y \mid x)$.

   b. The max-surprisal score is computed: $s_{\max} = \max_t(-\log P_\theta(y_t \mid x, y_{<t}))$.

   c. Both scores are compared against the CoRIM-sealed conformal thresholds.

   d. The conformity status (CONFORMING / NON_CONFORMING) is recorded.

   e. The ACI adaptive miscoverage level α_t is updated (Equation 65).

   f. The empirical coverage rate $\hat{C}(t)$ is updated over the sliding window (Equation 61).

5. **Drift entropy computation** (for multi-step generation).

   a. At generation step 0, the model's predicted distributions $P_k^{(0)}$ for future positions are cached (requires a forward pass over the target positions).

   b. At each subsequent generation step t, the current predictions $P_k^{(t)}$ are compared against the cached $P_k^{(0)}$.

   c. The drift entropy δ_t is accumulated: $\delta_t = \sum_k D_{KL}(P_k^{(0)} \| P_k^{(t)})$.

   d. The per-session drift entropy is tracked and compared against $\delta_{\max}$.

11.3.2 Periodic Audit Tasks

At intervals of $n_{\text{audit}} = 100$ monitoring epochs, the Drift Estimator performs additional computations that are too expensive for per-epoch execution:

- **Full-vocabulary KL divergence.** The accumulated Count-Min Sketch is compared against the CoRIM reference sketch using the full-vocabulary KL computation (Equation 26). This detects tail-distribution drift that Top-K monitoring would miss.
- **HyperLogLog support comparison.** The production HLL sketch is compared against the CoRIM reference HLL to detect vocabulary support expansion or contraction.
- **Bigram/trigram audit.** If bigram and trigram CMS sketches are configured, the divergence at these n-gram levels is computed.
- **KSWIN evaluation.** The KSWIN test (Section 7.4) is invoked with the current reference and test windows.

11.4 Subsystem 2: State Classifier

The State Classifier implements the compliance state machine (Section 6.4) and the detection pipeline decision logic (Section 7.5). It consumes metric values from the Drift Estimator and produces compliance state determinations.

11.4.1 Internal Components

Component 1: Detection Algorithm Bank. Maintains the running state of all three detection algorithms:

- **ADWIN instance.** Receives the per-epoch $D_{KL}^{\text{reported}}(t)$ value. Outputs `ADWIN_CHANGE` or `ADWIN_STABLE`.
- **Page-Hinkley instance.** Receives the per-epoch $D_{KL}^{\text{reported}}(t)$ value. Outputs `PH_ALARM` or `PH_STABLE`.
- **KSWIN instance.** Invoked on `ADWIN_CHANGE` or at the periodic audit interval. Outputs `KS_CONFIRMED` or `KS_NOT_CONFIRMED`.

Component 2: Threshold Comparator. Evaluates the direct threshold conditions:

- $D_{KL}^{\text{reported}}(t) > D_{\text{max}}$? (Rule R2)
- $D_{KL}^{\text{reported}}(t) > D_{\text{yellow}}$? (Rule R5)
- $\hat{C}(t) < C_{\text{alarm}}$? (Coverage violation)
- $\delta_t > \delta_{\text{max}}$? (Drift entropy violation, if configured)

Component 3: State Machine Engine. Implements the state transition rules T1–T5 (Section 6.4) as a deterministic finite automaton:

Listing 3: State Machine Engine Pseudocode

```
function update_state(current_state, metrics, algorithm_outputs):
    // Priority-ordered rule evaluation (Section 7.4.2)

    // R1: PH emergency brake
    if algorithm_outputs.PH == ALARM:
        return RED, "PH_ALARM"

    // R2: Direct threshold breach
    if metrics.drift_kl_reported > D_max:
        return RED, "THRESHOLD_BREACH"

    // R3: ADWIN + KSWIN confirmed drift
    if algorithm_outputs.ADWIN == CHANGE
       and algorithm_outputs.KSWIN == CONFIRMED:
        if metrics.drift_kl_reported > D_max:
            return RED, "CONFIRMED_BREACH"
        elif metrics.drift_kl_reported > D_yellow:
            return YELLOW, "CONFIRMED_WARNING"

    // R4: ADWIN change without KSWIN confirmation
    if algorithm_outputs.ADWIN == CHANGE
       and algorithm_outputs.KSWIN == NOT_CONFIRMED:
        log_soft_warning()
```

```
        elevate_sampling_rate()

    // R5: Warning threshold
    if metrics.drift_kl_reported > D_yellow:
        return YELLOW, "WARNING_THRESHOLD"

    // T4: Sustained warning check
    if current_state == YELLOW:
        if yellow_duration >= delta_t_crit:
            return RED, "SUSTAINED_WARNING"

    // T2/T5: Recovery check
    if current_state == YELLOW and recovery_count >= N_recovery:
        return GREEN, "RECOVERY"
    if current_state == RED
       and metrics.drift_kl_reported <= D_max
       and recovery_count >= N_recovery
       and root_cause_documented:
        return YELLOW, "RED_RECOVERY"

    // R6: Normal operation
    if metrics.drift_kl_reported <= D_yellow:
        if current_state == GREEN:
            return GREEN, "NOMINAL"
        else:
            increment_recovery_count()
            return current_state, "RECOVERING"
```

Component 4: Trend Projector. The Trend Projector provides forward-looking analysis by estimating when the drift-kl trajectory will breach the $D_{\max}$ threshold if the current trend continues.

Definition 11.1 (Projected Breach Time)**.** *Given the drift-kl trajectory* $\{D_{KL}(t_1), D_{KL}(t_2), \ldots, D_{KL}(t_n)\}$ *over the most recent* n *monitoring epochs, the projected breach time* t_{breach} *is:*

$$t_{breach} = t_n + \frac{D_{\max} - D_{KL}(t_n)}{\hat{\beta}_1} \tag{80}$$

where $\hat{\beta}_1$ *is the slope of the ordinary least squares regression of* D_{KL} *on epoch number:*

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(t_i - \bar{t})(D_{KL}(t_i) - \bar{D})}{\sum_{i=1}^{n}(t_i - \bar{t})^2} \tag{81}$$

The projected breach time is reported in the MAI-1 payload as an advisory field:

- $t_{\text{breach}} < 0$ or $\hat{\beta}_1 \leq 0$: Drift is decreasing or stable. No breach projected. Report `trend = STABLE`.
- $t_{\text{breach}} > 24$ hours: Drift is increasing slowly. Report `trend = SLOW_INCREASE` with projected breach time.
- $1 < t_{\text{breach}} \leq 24$ hours: Drift is increasing at a moderate rate. Report `trend = MODERATE_INCREASE` with projected breach time.
- $t_{\text{breach}} \leq 1$ hour: Drift is increasing rapidly. Report `trend = RAPID_INCREASE` with projected breach time. This triggers an automatic sampling rate elevation even if the current state is GREEN.

The Trend Projector provides the "early warning of the early warning": it alerts operators to developing drift trends before the threshold is breached, enabling proactive intervention rather than reactive response.

11.4.2 Conformal Coverage Integration

The State Classifier also evaluates the conformal coverage signal as a complementary state input:

Conformal Coverage State Integration

If the empirical conformal coverage $\hat{C}(t)$ drops below the alarm threshold C_{alarm} (Equation 62) while the drift-kl state is GREEN, the State Classifier **SHOULD** transition to YELLOW with reason `COVERAGE_VIOLATION`. This captures drift types (particularly covariate shift) that the drift-kl metric may miss but that degrade the model's per-prediction reliability.

If both the drift-kl state and the conformal coverage indicate alarm conditions simultaneously, the State Classifier **SHOULD** weight this as stronger evidence and reduce the sustained warning duration Δt_{crit} by 50% for the T4 transition (sustained YELLOW $\rightarrow$ RED). Corroborating signals from independent metrics increase confidence that genuine drift is occurring.

11.5 Subsystem 3: Intervention Controller

The Intervention Controller executes automated governance actions in response to compliance state transitions. It is the component that transforms monitoring from a passive observation activity into an active governance mechanism.

11.5.1 Intervention Policy Configuration

The Intervention Controller operates according to an *intervention policy*—a configuration document specifying which actions are triggered by which state transitions. The policy is deployment-specific and is set by the governance authority responsible for the model deployment.

Table 40: Default Intervention Policy for Clause AI-6

Trigger	Action	Description
GREEN → YELLOW	Elevate sampling	Increase monitoring sampling rate from 1% to 10%. Begin trend projection. Notify operations team via configured alert channel.
YELLOW → RED (T3: threshold breach)	Safe mode activation	Engage safe mode adapter (Section 11.5.2). Increase sampling to 100%. Generate incident report. Notify governance authority.
YELLOW → RED (T4: sustained)	Model routing	Route traffic to fallback model (Section 11.5.3). Preserve drifted model for diagnostic analysis. Generate incident report.
PH_ALARM (R1)	Emergency intervention	Immediately activate safe mode. Halt session that triggered alarm. Increase global sampling to 100%. Generate emergency incident report with full diagnostic payload.
Coverage violation	Temperature adjustment	Reduce generation temperature by $\Delta T = 0.1$ (Section 11.5.4). Elevate sampling. Log coverage violation in Decision Receipt chain.
RED → YELLOW (T5)	Gradual restoration	Reduce sampling from 100% to 10%. Maintain safe mode for N_{recovery} additional epochs. Monitor for relapse.
YELLOW → GREEN (T2)	Normal restoration	Return sampling to 1%. Deactivate safe mode. Log recovery event in Decision Receipt chain.

11.5.2 Safe Mode Adapter

The safe mode adapter is a lightweight intervention mechanism that constrains the model's output distribution without modifying its weights. When activated, the adapter applies the following constraints:

1. **Temperature reduction.** The generation temperature is reduced to $T_{\text{safe}} = 0.3$ (from the default operational temperature, typically 0.7–1.0). This concentrates the output distribution around the highest-probability tokens, reducing the probability of rare, potentially-hallucinated outputs.

2. **Top-K restriction.** The sampling is restricted to the top $K_{\text{safe}} = 50$ tokens (from the default $K = 200$–500 or unrestricted). This eliminates the tail of the distribution where drift-induced anomalous tokens are most likely to appear.

3. **Repetition penalty amplification.** The repetition penalty is increased to $r_{\text{safe}} = 1.5$

(from the default 1.0–1.2). This prevents inertial drift from locking the model into repetition loops.

4. **Maximum generation length restriction.** The maximum output length is reduced to $L_{\text{safe}} = 0.5 \times L_{\text{coherence}}$ (half the model's estimated Evans' Law coherence limit). This prevents the model from generating beyond the context length where coherence collapse is likely.

5. **System prompt injection.** A standardized safety prefix is prepended to the model's system prompt: "[SAFE MODE ACTIVE] The system is operating in a constrained mode due to detected behavioral drift. Responses may be shorter and more conservative than usual." This provides transparency to downstream consumers of the model's output.

Safe Mode Transparency

When safe mode is active, the MAI-1 attestation payload **MUST** include `intervention-status = SAFE_MODE` with the specific constraints applied (temperature, Top-K, repetition penalty, length limit). Consumers of the attestation payload can then make informed decisions about whether to accept the model's constrained output or route to an alternative system.

11.5.3 Model Routing

For deployments with access to multiple model instances, the Intervention Controller can route traffic away from the drifted model to a fallback:

1. **Hot standby routing.** Traffic is routed to a pre-validated standby model instance running the same architecture and CoRIM-validated configuration. The standby model has not been exposed to the production traffic that caused the primary model to drift. Latency impact: minimal (load balancer reconfiguration).

2. **Degraded capability routing.** Traffic is routed to a smaller, more constrained model that has been validated for the core use case but lacks the primary model's full capabilities. This is the "safe fallback" for deployments where no hot standby is available. Users experience reduced capability but maintained safety.

3. **Human-in-the-loop routing.** For the highest-risk deployments (medical, defense), the intervention can route model outputs through a human review queue before delivery to the end consumer. This adds latency but provides maximum safety assurance during the drift investigation period.

11.5.4 Temperature Adjustment

Temperature adjustment is the lightest-touch intervention, used when conformal coverage drops below the alarm threshold but drift-kl remains in the GREEN zone:

$$T_{\text{new}} = T_{\text{current}} - \Delta T \cdot \frac{C_{\text{alarm}} - \hat{C}(t)}{C_{\text{alarm}} - (1 - \alpha)} \tag{82}$$

where $\Delta T = 0.1$ is the maximum temperature reduction per adjustment step. The reduction is proportional to the severity of the coverage violation: a marginal violation produces a small reduction; a severe violation produces the full ΔT reduction.

The temperature adjustment is logged in the Decision Receipt chain and reported in the MAI-1 payload as `intervention-status = TEMPERATURE_ADJUSTED` with the adjustment magnitude.

11.6 Conditional Drift-KL Stratification

The standard drift-kl metric aggregates over all inputs. This can mask localized drift affecting specific input categories. Conditional stratification computes drift-kl separately for defined input strata:

$$D_{KL}^{c}(P_t \| P_0) = \sum_{y \in \mathcal{V}_K} P_t(y \mid c) \log \frac{P_t(y \mid c)}{P_0(y \mid c)} \tag{83}$$

where c is a stratum identifier (topic, language, task type, user demographic group).

11.6.1 Stratification Dimensions

Table 41: Drift-KL Stratification Dimensions

Dimension	**Strata**	**Governance Purpose**
Language	Per-language (EN, ZH, ES, ...)	Detects language-specific drift; prevents multilingual models from silently degrading on minority languages
Task type	Classification, generation, summarization, code, QA	Detects task-specific concept drift; a model may drift on code generation while remaining stable on text
Input length	Short ($<$ 100 tokens), medium (100–1,000), long ($>$ 1,000)	Detects context-length-dependent drift; related to Evans' Law coherence collapse
Topic/domain	Per-topic cluster (medical, legal, financial, general)	Detects domain-specific drift; critical for sector-specific deployments
Demographic group	Per-group (if demographic metadata available)	Detects fairness-relevant drift; required by NIST AI RMF MEASURE 2.11

11.6.2 Implementation

Stratified drift-kl maintains separate Count-Min Sketches per stratum. The stratum assignment is performed by a lightweight classifier operating on the input embedding (not on the output), ensuring that the stratification does not depend on the model's (potentially drifted) output.

Stratified Monitoring for High-Risk Deployments

Deployments classified as high-risk under the EU AI Act Annex III or as Class II/III under FDA SaMD guidance **SHOULD** implement at least two stratification dimensions (language and task type at minimum). The per-stratum drift-kl values **SHOULD** be reported in the MAI-1 payload as extension fields. A per-stratum breach ($D^c_{KL}(t) > D^c_{\max}$) **SHOULD** trigger a YELLOW state transition even if the aggregate drift-kl remains below D_{yellow}.

11.7 Feedback Loops: Closing Monitoring Debt

Sculley et al. (2015) identified "monitoring debt" as a critical form of technical debt in ML systems: the failure to implement monitoring creates a hidden liability that compounds over time. Clause AI-6 closes this debt by establishing explicit feedback loops between monitoring outputs and operational decisions.

11.7.1 Feedback Loop 1: Monitoring → Baseline Refresh

When the Drift Monitor detects sustained drift that triggers a Sentinel Protocol baseline update (Section 5.4), the monitoring system feeds the drift characteristics into the baseline construction process:

- The nature of the detected drift (which shift type, which strata affected) informs the validation set composition for the new baseline.
- The magnitude of the detected drift informs the governance risk margin Δ_{gov} for the new threshold: if the previous margin was too tight (excessive false positives), the new margin is widened; if too loose (missed genuine drift), the new margin is tightened.
- The detection latency (how long between drift onset and detection) informs the window configuration for the new monitoring cycle.

11.7.2 Feedback Loop 2: Monitoring → Model Improvement

The drift characteristics provide actionable intelligence for model development:

- **Persistent prior probability shift** (e.g., RLHF token frequency distortion) indicates that the alignment training process introduced distributional artifacts. The model development team can use the specific token frequency deviations identified by drift-kl to refine the RLHF reward function.
- **Recurrent covariate shift** (detected by conformal coverage degradation) indicates that the production input distribution has diverged from the training distribution. The model development team can use the input strata most affected by drift to guide data collection for the next training cycle.
- **Evans' Law coherence collapse** (detected by drift entropy and windowed drift-kl near $L_{\text{coherence}}$) indicates that the model's effective context window is shorter than its architectural context window. The model development team can prioritize architectural improvements (e.g., better positional encoding, sparse attention) for the affected context length range.

11.7.3 Feedback Loop 3: Monitoring → Insurance Actuarial Update

The conformal coverage rate and drift-kl trajectory provide continuous inputs for actuarial model updates:

- If the empirical conformal coverage consistently exceeds $(1 - \alpha)$ (the model is better than the bound), the insurer can tighten the pricing—the risk is lower than assumed.
- If the empirical coverage approaches the alarm threshold, the insurer can issue a premium adjustment or require operational changes before the next policy renewal.
- The cumulative baseline drift $D_{\text{cumulative}}$ (Equation 29) across the policy period provides a single scalar summary of the model's behavioral stability for actuarial reporting.

11.8 MAI-1 Layer 2 Payload: Drift Fields

The Drift Monitor's output is encoded in the MAI-1 Layer 2 attestation payload. The following fields are defined for Clause AI-6:

Listing 4: MAI-1 Layer 2 Payload: Clause AI-6 Fields

```
"invariants": {
    "drift-kl": {
        "value": 0.087,                    // current drift-kl (nats)
        "window-sizes": [1000, 50000, 500000],
        "per-window": [0.042, 0.071, 0.087],
        "baseline-corim": "urn:corim:drift-ref-v3",
        "estimation-method": "top-k-truncated",
        "k": 100,
        "sampling-rate": 0.01
    },
    "drift-kl-max": 0.171,                 // certified threshold (nats)
    "drift-entropy": {                      // optional extension
        "value": 0.83,
        "max": 2.0,
        "status": "NOMINAL"
    }
},
"thresholds": {
    "drift-kl-max": 0.171,
    "drift-kl-yellow": 0.103,
    "drift-entropy-max": 2.0
},
"compliance-status": 1,                    // 0=GREEN, 1=YELLOW, 2=RED
"breached-invariant": null,                // null if no breach; "drift-
   kl" if breached
"monitoring-status": "ACTIVE",             // ACTIVE, SAFE_MODE,
   UNAVAILABLE
"intervention-status": null,               // null, SAFE_MODE,
   TEMPERATURE_ADJUSTED, ROUTED
"trend": {
    "direction": "SLOW_INCREASE",
    "projected-breach-hours": 18.4,
    "slope-nats-per-epoch": 0.00012
},
"conformal": {
    "score-type": "nll",
    "threshold": 4.82,
    "target-coverage": 0.90,
```

```
    "empirical-coverage": 0.88,
    "coverage-status": "WARNING",
    "mean-set-width": 127.3,
    "set-width-trend": 0.034,
    "aci-alpha-t": 0.092,
    "calibration-corim": "urn:corim:ai6-cp-cal-v1"
},
"detection-algorithms": {
    "adwin": "STABLE",
    "page-hinkley": "STABLE",
    "kswin-last-result": "NOT_CONFIRMED",
    "kswin-last-epoch": 4200
},
"stratified": {                          // optional extension
    "en": {"drift-kl": 0.065, "status": "GREEN"},
    "zh": {"drift-kl": 0.142, "status": "YELLOW"},
    "code": {"drift-kl": 0.091, "status": "GREEN"}
},
"sentinel": {
    "baseline-version": 3,
    "parent-corim": "urn:corim:drift-ref-v2",
    "cumulative-drift": 0.23,
    "last-update-reason": "FINE_TUNING",
    "last-update-date": "2026-01-15T14:30:00Z"
}
```

This payload is self-authorizing: an auditor, regulator, insurer, or procurement officer can inspect the payload and determine the model's drift status, the monitoring methodology, the threshold calibration, the detection algorithm states, the conformal coverage, and the baseline provenance—without contacting the model vendor, accessing the model's internals, or running any additional tests.

Honest Framing

What the implementation architecture provides: A complete, production-ready reference architecture for Clause AI-6 monitoring, with three independent subsystems (measurement, classification, intervention), automated governance actions, conditional stratification for localized drift detection, feedback loops that close monitoring debt, and a self-authorizing MAI-1 attestation payload containing all information needed for independent compliance assessment.

What the implementation architecture does not provide: A turnkey software implementation. This section specifies the architecture, data flows, algorithms, and interfaces—not the source code. The architecture is implementation-language-agnostic and hardware-platform-agnostic by design: it can be realized in Python, C++, Rust, or any language with access to the model's logit outputs and hidden states. The computational feasibility analysis (Section 9) demonstrates that the architecture fits within production latency and memory constraints, but the actual performance depends on implementation quality, hardware characteristics, and deployment-specific factors that are outside the scope of a governance specification.

12 Differentiation: What No Other Framework Provides

The distribution drift monitoring space is not empty. Commercial observability platforms, academic research frameworks, and internal monitoring tools at frontier AI companies all address

aspects of the drift detection problem. This section positions Clause AI-6 within this landscape by identifying the specific combination of properties that distinguishes the Auburn Governance Stack's approach from every existing alternative.

12.1 The Observability–Governance Gap

The dominant paradigm in production ML monitoring is *observability*: dashboards that display metrics, alert when thresholds are crossed, and provide diagnostic tools for investigation. Leading platforms in this space include Evidently AI, WhyLabs, Arize AI, Fiddler AI, and NannyML. These platforms provide valuable operational tooling. They do not provide governance infrastructure.

The distinction is precise:

Table 42: Observability vs. Governance: Structural Comparison

Property	**Observability (Evidently, WhyLabs, etc.)**	**Governance (Clause AI-6)**
Metric computation	Yes—KL divergence, PSI, Jensen-Shannon, Wasserstein	Yes—KL divergence with formal estimation methods (kNN, Donsker-Varadhan, surrogate)
Threshold calibration	Ad hoc or user-configured; no formal methodology	Conformal calibration with distribution-free false positive rate guarantee (Section 6.3.2)
Baseline integrity	Stored in platform database; mutable; no tamper evidence	Cryptographically sealed in CoRIM artifact; COSE-signed; SCITT-registered; immutable
Baseline versioning	Manual or automated replacement; no anti-drift-washing protection	Reilly Sentinel Protocol with SEP documentation, cross-reference drift tracking, and immutable provenance chain (Section 5.4)
Compliance determination	Alert-based; configurable severity levels; interpretive discretion	Binary pass/fail; deterministic state machine; no interpretive wiggle room
Automated intervention	Alert routing; webhook triggers; no standardized intervention protocol	Safe mode adapter, model routing, temperature adjustment with standardized intervention policy (Section 11.5)
Per-prediction uncertainty	Not provided	Conformal prediction with distribution-free coverage guarantee (Section 8)
Insurability bridge	Not provided	CP-bounded failure probability enables actuarial pricing (Section 8.6)
Regulatory mapping	Generic compliance dashboards; no specification-level mapping	Citation-level mapping to EU AI Act, SR 11-7, FDA PCCP, NIST RMF, SEC 15c3-5, Solvency II (Section 10)
Interoperability	Proprietary APIs; platform lock-in	IETF standards (CoRIM, RATS, SCITT); open attestation format (MAI-1)

The gap is not in the metrics—observability platforms compute the same divergence measures. The gap is in the *infrastructure around the metrics*: the cryptographic integrity, the formal calibration, the binary compliance, the automated intervention, the per-prediction uncertainty,

and the regulatory mapping that transform a monitoring metric into a governance artifact.

12.2 The Five Unique Properties

No other framework—academic, commercial, or internal—combines all five of the following properties:

Property 1: Cryptographic baseline binding (CoRIM). The reference distribution is not stored in a database that can be silently modified. It is sealed in a COSE-signed CoRIM artifact, registered in a SCITT append-only ledger, and linked to the model's identity and deployment configuration. Any modification to the baseline is a cryptographic event that produces a new artifact with a new signature, linked to its predecessor by the Sentinel Protocol. No existing drift monitoring tool provides this level of baseline integrity.

Property 2: Conformal prediction integration. The combination of aggregate drift monitoring (drift-kl) with per-prediction uncertainty quantification (conformal prediction) is unique. Observability platforms provide drift metrics but cannot bound the probability of individual prediction failure. Academic conformal prediction research provides per-prediction bounds but does not integrate them with continuous drift monitoring or cryptographic baselines. Clause AI-6 unifies both into a single, coherent framework.

Property 3: Binary compliance with deterministic state machine. The compliance determination is deterministic: given the same drift-kl signal, two independent implementations of the Clause AI-6 state machine must produce identical state sequences. This is a design decision borrowed from protocol engineering (TCP state machines are deterministic) and is essential for regulatory enforcement. A compliance determination that depends on operator judgment, configuration choices, or platform-specific behavior is not enforceable by a regulator. Clause AI-6's determinism ensures that compliance can be independently verified.

Property 4: Formal detection algorithm composition with characterized properties. The three-algorithm pipeline (ADWIN, Page-Hinkley, KSWIN) is not an ad hoc collection of detectors. Each algorithm was selected against formal criteria (Section 7), assigned to a specific drift type based on its theoretical properties, and composed with formally characterized false alarm rates and detection delays (Propositions 7.3 and 7.4). The composition is documented at a level sufficient for independent reproduction and formal verification.

Property 5: The honest framing. Every section of this document includes an explicit "Honest Framing" statement documenting what the framework does not provide. No other governance specification in the AI space—proprietary or academic—maintains this level of epistemic honesty. The honest framing is not a weakness; it is a structural advantage. A framework that overpromises is discredited when its limitations are discovered; a framework that documents its limitations from the outset builds the trust necessary for regulatory adoption.

12.3 Comparison with Academic Drift Detection Research

The academic literature on drift detection (Lu et al., 2018; Gama et al., 2014; Webb et al., 2016) provides the theoretical foundations upon which Clause AI-6 builds. The contribution of Clause AI-6 is not the invention of new detection algorithms but the *operationalization* of known algorithms into a governance-grade specification:

- Academic drift detection research typically evaluates algorithms on synthetic or benchmark datasets. Clause AI-6 specifies calibrated parameters for production LLM deployments with vocabulary sizes of 100,000+ tokens and hidden state dimensions of 4,096+.
- Academic research evaluates algorithms in isolation. Clause AI-6 specifies a composed pipeline with formally characterized joint properties.
- Academic research does not address baseline integrity, regulatory compliance, or commercial insurability. Clause AI-6 integrates these concerns into the core specification.

The Auburn Governance Stack does not compete with academic research; it *consumes* academic research and transforms it into deployable governance infrastructure.

12.4 Comparison with Frontier AI Company Internal Monitoring

Frontier AI companies (Anthropic, OpenAI, Google DeepMind) maintain internal monitoring systems that are not publicly documented. Based on published safety frameworks and researcher presentations, these systems likely include:

- Output distribution monitoring during training and evaluation.
- Automated evaluation suites (benchmarks, red-teaming).
- Human feedback loops for detecting production issues.

Clause AI-6 differs from these internal systems in two structural ways:

1. **Transparency.** Internal monitoring systems are proprietary and undocumented. Clause AI-6 is a public specification that can be independently implemented, verified, and audited. The governance value of monitoring depends on its verifiability by third parties—an internal system that cannot be inspected by a regulator provides limited governance assurance.
2. **Standardization.** Internal monitoring systems are bespoke to each company's infrastructure. Clause AI-6 provides a common standard that enables cross-vendor comparison: an auditor can compare the drift-kl compliance status of a model from Vendor A against a model from Vendor B using the same metric, the same threshold calibration methodology, and the same compliance state machine. This standardization is the precondition for regulatory enforcement.

13 Anticipated Objections and Resolutions

A governance specification that cannot withstand critical scrutiny is not ready for deployment. This section enumerates the five most likely objections to Clause AI-6 and provides the technical or structural resolution for each.

13.1 Objection 1: "The drift-kl signal is too noisy for reliable governance."

Statement. The KL divergence estimated from a finite sample of tokens is inherently noisy. Natural variation in user prompts, session-to-session variance, and the stochastic nature of autoregressive generation produce drift-kl fluctuations that may be mistaken for genuine drift.

Resolution. This objection is valid for naive implementations that apply a fixed threshold to a single-window KL estimate. Clause AI-6 addresses noise through four mechanisms:

1. **Multi-scale windowing** (Section 6.2). The slow window ($W_{\text{slow}} = 500{,}000$ tokens) averages over sufficient data to reduce sampling variance below the drift resolution threshold. The fast window ($W_{\text{fast}} = 1{,}000$ tokens) is explicitly acknowledged as noisy and is processed by the Page-Hinkley test, which has its own false alarm control.

2. **Conformal threshold calibration** (Section 6.3.2). The threshold $D_{\max}$ is derived from the model's own empirical null distribution—the distribution of drift-kl values under no-drift conditions. The threshold is set at the $(1 - \alpha_{\text{FP}})$-quantile of this null distribution, providing a formal false positive rate guarantee.

3. **ADWIN's adaptive windowing** (Section 7.2). ADWIN automatically adjusts its window size to the noise level: during stable periods, the window grows, accumulating statistical power; during drift events, the window shrinks, capturing the change quickly. This adaptation is governed by Theorem 7.1, which bounds the false alarm probability.

4. **KSWIN confirmation** (Section 7.4). ADWIN detections are confirmed by the independent KSWIN test before triggering a compliance state transition (Rule R3, Section 7.5). This two-stage confirmation reduces the effective false alarm rate to the product of the individual rates.

The combined pipeline false alarm rate for a RED state transition is bounded by Proposition 7.3: $< 0.01\%$ per monitoring epoch under typical configurations. This is comparable to the false alarm rates of established industrial safety systems (fire alarms, industrial process monitors).

13.2 Objection 2: "The baseline becomes stale as the world changes."

Statement. The reference distribution P_0 is fixed at validation time. As the real world changes (new events, evolving language, shifting user demographics), the production distribution P_t will naturally diverge from P_0 even if the model is functioning correctly. Over time, the baseline becomes stale, producing persistent Yellow or Red states that do not reflect genuine model degradation.

Resolution. This is the most sophisticated objection and the reason the Reilly Sentinel Protocol exists (Section 5.4). The resolution operates at three levels:

1. **Governance risk margin.** The threshold $D_{\max}$ includes a governance risk margin Δ_{gov} (Section 6.3) that provides headroom for natural environmental drift. The margin is calibrated by deployment context: research deployments with highly variable input distributions receive wider margins than medical devices with constrained input domains.

2. **Scheduled baseline refresh.** The Sentinel Protocol defines `SCHEDULED_REFRESH` as a legitimate update reason. Governance authorities can establish a baseline refresh cadence (e.g., quarterly) with abbreviated SEP requirements, acknowledging that the world has changed and the baseline should reflect current conditions.

3. **Anti-drift-washing protection.** The Sentinel Protocol's cross-reference drift tracking (Equation 29) ensures that repeated baseline refreshes do not silently transform the model's behavioral envelope. The cumulative drift $D_{\text{cumulative}}$ across all baseline updates is tracked, and if it exceeds $D_{\text{lifecycle}} = 3 \times D_{\max}$, governance escalation is triggered. This means the model can adapt to changing conditions through managed baseline updates, but the total adaptation is bounded and auditable.

The analogy to financial auditing is precise: a company's financial statements are audited annually against current accounting standards, not against the standards from the year the company was founded. The standards evolve (baseline refresh), but each transition is documented and the cumulative change is tracked. Clause AI-6 applies the same principle to model behavior.

13.3 Objection 3: "The monitoring overhead is unacceptable for production systems."

Statement. Any monitoring infrastructure that adds latency to the inference pipeline will be rejected by production engineering teams optimizing for throughput and response time.

Resolution. Section 9 provides the complete computational tractability analysis. The key results:

- Under normal conditions (GREEN state, 1% sampling): 0.15 μs per generated token, or 0.0005% of inference latency for a 70B model. This is three orders of magnitude below the 1% budget.
- Under worst-case conditions (RED state, 100% sampling): 14.4 μs per generated token, or 0.05% of inference latency. Still 20× below the 1% budget.
- Memory footprint: 56 MB total, with < 5 MB on-GPU (0.004% of a 70B model's VRAM).

The monitoring overhead is not "acceptable"; it is *imperceptible*. The sidecar architecture ensures process isolation: even if the monitoring system experiences a performance degradation, the inference pipeline is unaffected.

The objection often reflects a conflation of "monitoring" with "evaluation." Running a benchmark suite against the model is expensive. Computing a per-token KL divergence contribution using Top-K truncation and asynchronous sampling is not.

13.4 Objection 4: "False positives will create alarm fatigue and operational resistance."

Statement. If the drift monitoring system produces frequent false alarms, operations teams will learn to ignore it, reducing the system to a noise generator that provides no governance value.

Resolution. Alarm fatigue is a genuine and well-documented risk in safety-critical systems (medical alarms, industrial SCADA systems). Clause AI-6 mitigates this risk through five mechanisms:

1. **Formal false positive rate control.** The conformal threshold calibration (Section 6.3.2) provides a mathematical guarantee on the false positive rate. For $\alpha_{\text{FP}} = 0.01$, no more than 1% of monitoring epochs produce false alarms under null conditions.
2. **Two-stage confirmation.** ADWIN detections require KSWIN confirmation before triggering a state transition (Rule R3). This reduces the effective false alarm rate to $\delta_{\text{ADWIN}} \times \alpha_{\text{KS}} \approx 0.002 \times 0.01 = 0.00002$—one false alarm per 50,000 monitoring epochs.
3. **Graduated response.** The GREEN → YELLOW → RED progression provides a graduated response. A single threshold exceedance produces a YELLOW warning, not a RED alarm. The YELLOW state increases monitoring intensity and triggers investigation, but does not activate safe mode or model routing. Only sustained or severe drift produces a RED alarm. This ensures that the most disruptive interventions are reserved for genuine events.

4. **Recovery hysteresis.** The $N_{\text{recovery}} = 5$ consecutive in-bound epochs required for state recovery (transitions T2, T5) prevents oscillation between states due to noise. A system at the boundary between GREEN and YELLOW does not flip-flop on every epoch; it stabilizes in one state or the other.

5. **Empirical jitter calibration.** The threshold calibration procedure (Section 6.3.1) explicitly measures the model's natural jitter and sets the threshold above it. The empirical null distribution captures deployment-specific noise characteristics that a theoretical analysis would miss.

13.5 Objection 5: "KL divergence in high dimensions is unreliable."

Statement. The curse of dimensionality (Section 4.2) renders KL divergence estimation unreliable in the high-dimensional hidden state spaces of modern LLMs. The kNN estimator has high variance; the Donsker-Varadhan bound may be loose; the variational surrogate is trained on the validation distribution and may produce unreliable estimates for out-of-distribution states.

Resolution. This objection is technically correct for the hidden state monitoring component. Clause AI-6 addresses it through architectural design rather than by solving the fundamental statistical problem:

1. **The mandatory invariant operates in the output space, not the hidden state space.** The `drift-kl` field in the MAI-1 payload is computed on the output token distribution—a categorical distribution over $|V|$ states where KL divergence is exact and tractable. Hidden state monitoring is a supplementary signal, not the mandatory invariant.

2. **JL projection reduces dimensionality with formal guarantees.** The Johnson-Lindenstrauss projection (Section 9.2) reduces the hidden state dimension from $d = 8{,}192$ to $d' = 128$ with a formal 30% distortion bound. In the projected space, density estimation methods are significantly more reliable.

3. **The two-tier architecture provides cross-validation.** The variational surrogate provides a fast initial estimate; the Donsker-Varadhan drift head confirms it. If the two estimates disagree significantly, the system logs a diagnostic event and falls back to output-space-only monitoring. The hidden state monitoring degrades gracefully rather than producing unreliable governance signals.

4. **The conformal prediction integration provides a non-parametric alternative.** For deployments where hidden state KL estimation is deemed unreliable, the conformal coverage monitor (Section 8.3) provides drift detection without any density estimation at all. Coverage monitoring requires only a scalar comparison per prediction, with no high-dimensional computation.

The honest response to this objection is: *yes, high-dimensional KL estimation is hard, and Clause AI-6 does not pretend otherwise.* The specification addresses the difficulty through dimensionality reduction, multi-method cross-validation, graceful degradation, and a complementary non-parametric detection pathway. The mandatory invariant avoids the problem entirely by operating in the tractable output space.

14 Conclusion

14.1 The Problem Restated

When a foundation model is validated, it produces a specific distribution of outputs. When it is deployed, the distribution changes. The change may be benign (natural variation in user inputs) or pathological (coherence collapse, adversarial manipulation, catastrophic forgetting). Without continuous monitoring, there is no way to distinguish between these cases—and no evidence to present to a regulator, insurer, or auditor when they ask: "Is this model still behaving like the model you validated?"

This is the behavioral envelope problem, and it is the single largest obstacle to regulated AI deployment. Static benchmarks prove nothing about operational behavior. Periodic evaluations sample behavior at checkpoints but miss the drift between them. Observability dashboards display metrics but provide no cryptographic integrity, no formal calibration, no binary compliance determination, and no bridge to commercial insurance.

14.2 The Contribution of Clause AI-6

Clause AI-6 resolves the behavioral envelope problem by specifying the complete technical infrastructure for continuous distribution drift monitoring within the Auburn Governance Stack's MAI-1 attestation architecture. The specification provides:

1. **A mandatory invariant** (Equation 30): $D_{KL}(P_t \| P_0) \leq D_{\max}$ at every attested measurement point, with the reference distribution P_0 cryptographically sealed in a CoRIM artifact and the threshold $D_{\max}$ derived from the model's own empirical null distribution using distribution-free conformal calibration.

2. **A formal taxonomy** (Section 3): the Quinonero-Candela dataset shift classification adapted for LLMs, with explicit characterization of which shift types drift-kl detects and which require complementary metrics.

3. **A complete estimation methodology** (Section 4): information-theoretic foundations, high-dimensional estimation methods (kNN, Donsker-Varadhan, variational surrogate), alternative divergence measures with selection justification, and formal approximation error bounds.

4. **A cryptographic baseline infrastructure** (Section 5): three reference distribution strategies, efficient sketch representations (Count-Min Sketch, HyperLogLog), a CDDL-specified CoRIM extension, and the Reilly Sentinel Protocol for anti-drift-washing baseline versioning.

5. **A calibrated threshold methodology** (Section 6): multi-scale windowing with statistical reliability derivation, conformal threshold calibration, governance risk margins by deployment context, and a deterministic compliance state machine with six transition rules.

6. **A composed detection pipeline** (Section 7): three algorithms (ADWIN, Page-Hinkley, KSWIN) with formally characterized false alarm rates and detection delays, priority-ordered decision logic, and session-to-global aggregation.

7. **A conformal prediction integration** (Section 8): distribution-free per-prediction coverage guarantees, the exchangeability-drift linkage for independent drift detection, adaptive conformal inference for non-stationary environments, and the bridge to commercial insurability via Munich Re *aiSure* requirement mapping.

8. **A computational tractability proof** (Section 9): four optimization techniques (JL projection, Top-K truncation, asynchronous sampling, variational surrogates) demonstrating $< 0.05\%$ latency overhead under worst-case conditions and 56 MB total memory footprint.

9. **A six-framework regulatory mapping** (Section 10): citation-level mapping to EU AI Act, SR 11-7, FDA PCCP, NIST AI RMF, SEC 15c3-5, and Solvency II, with frontier safety framework gap analysis and regulatory pressure timeline.

10. **A production-ready implementation architecture** (Section 11): the Drift Monitor with three subsystems (Estimator, Classifier, Controller), automated intervention mechanisms (safe mode, model routing, temperature adjustment), conditional stratification, feedback loops, and a self-authorizing MAI-1 Layer 2 attestation payload.

14.3 The Governance Consequence

With Clause AI-6, distribution drift becomes an *auditable property*. A regulator can examine the MAI-1 attestation payload and determine whether the model's behavioral envelope was maintained. An insurer can price coverage based on the conformal failure probability bound. A procurement officer can require Clause AI-6 conformance in an RFP. A model vendor can demonstrate that their monitoring infrastructure meets the highest available standard.

With Clause AI-6, distribution drift becomes an *insurable property*. The conformal prediction integration provides the distribution-free, per-prediction failure probability bounds that actuarial models require. The CoRIM-sealed baseline provides the verifiable reference point that claims assessment requires. The compliance state machine provides the binary determination that policy terms require.

With Clause AI-6, distribution drift becomes an *enforceable property*. The binary compliance determination—pass or fail, GREEN or RED, conforming or non-conforming—eliminates the interpretive discretion that undermines enforcement. A system that claims Clause AI-6 conformance either satisfies the invariant or it does not. There is no partial credit.

14.4 What Remains

Clause AI-6 is the fourth mandatory invariant in the MAI-1 specification, joining AI-8 (entropy floor), AI-2 (gradient stability), and AI-4 (thermal integrity). The fifth mandatory invariant—AI-7 (structural coherence)—completes the set. Together, the five invariants provide continuous health monitoring across the full spectrum of model pathologies: diversity collapse (AI-8), training instability (AI-2), hardware compromise (AI-4), behavioral drift (AI-6), and structural degradation (AI-7).

The Auburn Governance Stack's enforcement layer (CTS-1) and application layer (sector profiles) consume the evidence produced by these invariants. Clause AI-6's drift monitoring evidence feeds directly into:

- The CTS-1 conformance test suite (assertions AS-SM-L2-10 through TE-SM-L2-10.04).
- The EU AI Act compliance profile (Articles 15, 9, 72).
- The FDA SaMD sector profile (PCCP drift monitoring).
- The financial services sector profile (SR 11-7 PSI alignment).
- The insurance underwriting package (conformal coverage, actuarial inputs).

The behavioral envelope is now defined, measured, calibrated, monitored, attested, and enforceable. The model's drift is no longer a silent risk accumulating in the dark. It is a governed property of a governed system.

Intellectual Property (IP) Declaration

The methods, logic structures, and "Certified Constant" registries contained in this document are the sole property of Ryan Fields.

Public License (Non-Commercial)

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) license.

- **Academic Use:** Researchers may share and use this framework for non-commercial academic purposes, provided full attribution is given to Ryan Fields.
- **No Derivatives:** No modifications or "remixes" of the "Certified Constants" or logical proofs are permitted without express written consent.

Commercial Prohibition

Commercial use of this framework is strictly prohibited. This includes, but is not limited to:

- Use within proprietary high-frequency trading (HFT) risk models.
- Integration into commercial high-assurance AI governance software.
- Use by private financial institutions for "tail-risk" auditing of prime distribution variance.

UncleBroFields@proton.me
fieldsryanchristopher@gmail.com

Auburn Patent Family Fields
Clause AI-6: Distribution Drift Bound
Ryan Fields