# LoopGuard AI: Technical Architecture & Metrics

Deep-Dive into the Active Governance Layer for Autonomous Agents

**RATIUM.AI | ENGINEERING RESEARCH DIVISION | REFERENCE ARCHITECTURE V2.1**

## 1. Architectural Philosophy: The Governance Shim

LoopGuard AI is architected as a high-performance, model-agnostic **Interception Shim**. In the current AI landscape, governance is often treated as an asynchronous auditing process. LoopGuard AI moves governance to the *critical path*, functioning as a real-time deterministic gatekeeper that sits between the LLM (Large Language Model) and the operational environment.

> **Core Thesis:** The "Reasoning-Realization Gap" necessitates a system that decouples linguistic generation from operational decision-making. LoopGuard AI provides this decoupling through a strict *Shell/Core Discontinuity*.

### Shell/Core Discontinuity

The **Core** (the LLM) remains a probabilistic, non-deterministic engine optimized for high-dimensional semantic mapping. The **Shell** (LoopGuard AI) is a deterministic policy-enforcement layer. This allows organizations to leverage the creative power of SOTA models while maintaining a "Zero-Trust" posture regarding the model's raw output.

## 2. The Decision State Machine (DSM)

At the heart of the architecture is the Decision State Machine. Every token stream emitted by the Core is intercepted, buffered, and analyzed against a multidimensional policy matrix before being committed to the application state.

### Primary Operational States

- **SHIP:** The response meets all structural, safety, and logical consistency thresholds. Immediate release to user/environment.
- **HOLD:** Detected ambiguity or low NFCI score. The output is quarantined for a second-pass automated review or human intervention.
- **RESTRICT:** Triggers a safe-mode fallback. The original response is suppressed and replaced by a pre-validated, policy-aligned template.
- **ROLLBACK:** High-risk deviation detected. The agent's session is terminated, and the environment state is reverted to the last known-safe checkpoint.

# 3. The NFCI Metric: Measuring Logical Coherence

Traditional evaluation metrics (BLEU, ROUGE, or simple sentiment) fail to capture the *reasoning integrity* of a response. LoopGuard AI introduces the **Non-Formal Consistency Index (NFCI)**.

## Metric Dimensions

The NFCI is calculated based on three primary vectors:

1. **Structural Grounding:** Measuring the logical distance between a claim and its internal supporting evidence within the response.
2. **Stability Variance:** A measurement of consistency across multiple latent samplings of the model's reasoning path.
3. **Policy Compliance Vector:** Real-time mapping against formal organizational constraints and safety boundaries.

```
// Concept Definition: NFCI Score Logic NFCI = (G * w1) + (S * w2) + (C * w3)
Where: G = Grounding Consistency S = Stability / Variance delta C =
Constraint Alignment w = Weights based on operational risk profile
```

# 4. The Evidence Bundle Protocol

In regulated industries (Finance, Healthcare, Defense), "Black Box" AI decisions are unacceptable. LoopGuard AI generates a cryptographically signed **Evidence Bundle** for every significant action.

## Bundle Contents

- **Trace Data:** The complete chain of prompts and outputs (Core state).
- **Evaluation Signals:** Raw scores from the NFCI and other internal heuristics.
- **Decision Justification:** A machine-readable log explaining why a specific Gate (e.g., HOLD) was triggered.

- **Timestamp & Versioning:** Verification of the specific policy version active at the time of the decision.

## 5. Deployment & Integration Patterns

Designed for enterprise-scale flexibility, LoopGuard AI supports three primary integration patterns:

### Pattern A: The SDK / Library (Embedded)

The shim is integrated directly into the application code as a middleware layer. Best for low-latency, mission-critical agents where the overhead must be minimized.

### Pattern B: The Governance Proxy (Networked)

LoopGuard AI acts as an API Gateway. All LLM calls pass through the proxy, which enforces the policy before returning the response to the caller.

### Pattern C: The On-Premises Orchestrator

For data-sovereign environments, the system runs alongside self-hosted models, providing a localized governance layer that never leaves the private cloud.

---