

The Deterministic Data Plane:

Transitioning Governance from Passive Evaluation to Inline Structural Constraints

Rob Caswell

Authority Engineering

Technical Report Series on Derived Law and Autonomous Operations

TR-005

July 2026

Publication Record

The Deterministic Data Plane: Transitioning Governance from Passive Evaluation to Inline Structural Constraints

Author: Rob Caswell

Publisher: Cyber Terrain LLC

Published: 2026-07-01

Zenodo version: v1.0

DOI: <https://doi.org/10.5281/zenodo.21742412>

Website: <https://cyberterrain.ai/papers/tr-005-deterministic-data-plane.pdf>

Corpus Context

Corpus: Authority Engineering

Publication series: Technical Report Series on Derived Law and Autonomous Operations

Publication designation: TR-005

Document ID: TR-005

Object family: TR-005

Corpus document ID: AE-TR-0005

Publisher-imprint edition prepared for versioned republication.

The prior Zenodo record remains the citable record until its new version is published.

Abstract

Modern enterprise governance architectures are fundamentally detached from the execution vectors they seek to regulate. By separating the logical plane of policy evaluation from the physical plane of state mutation, contemporary security controls introduce fatal control-path latency, leaving them structurally incapable of preventing unauthorized consequence. Authority Engineering establishes that valid governance cannot be retrospective; rather, authority must be locally owned, bounded, and attached directly to the consequence seam.

This report establishes the immutable engineering requirements dictated by these canonical constraints. Part I derives these requirements directly from the accepted corpus, independent of any specific platform or software vendor. Part II details a real-world validation of these patterns, analyzes legacy structural vulnerabilities, and presents a singular, concrete engineering realization that satisfies these constraints through an execution-time structural enforcement architecture—defining an operational framework known as the Deterministic Data Plane.

PART I: CORPUS-DERIVED ENGINEERING REQUIREMENTS

Scope and Applicability

This report establishes an engineering specification for systems requiring deterministic governance over consequence-bearing actions. To maintain clear boundaries of inquiry, this report does not claim:

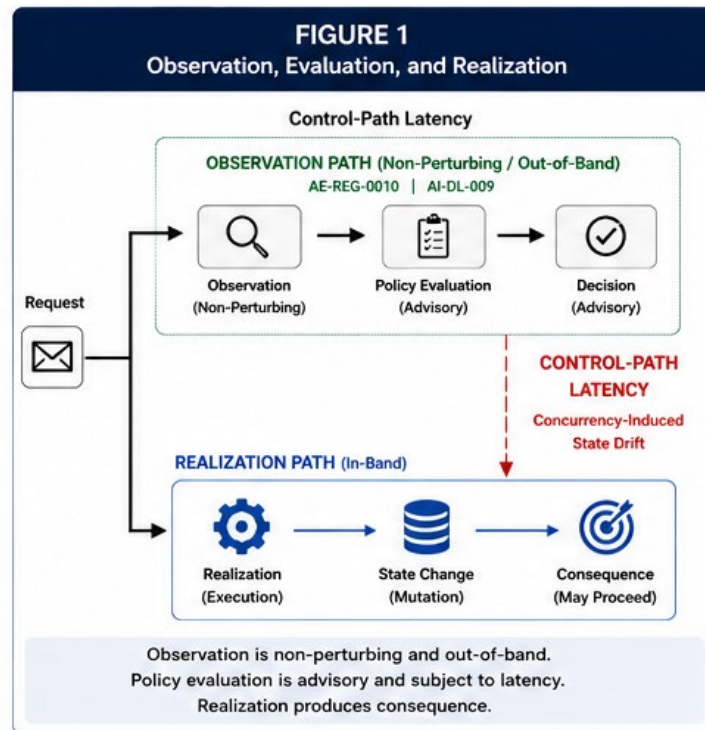
- That every enterprise governance problem requires inline structural enforcement.
- That all decoupled policy engines are inherently ineffective for non-consequence-bearing states.
- That retrospective controls, log aggregation, or post-incident auditing are without value.
- That threat observation, anomaly detection, or out-of-band incident response become unnecessary.

Passive, retrospective, and advisory security architectures remain highly effective for forensic analysis, behavioral profiling, and post-hoc threat hunting. This report addresses a distinct problem space: the deterministic regulation of state mutations at the point of irreversible realization. The requirements and realizations derived herein apply strictly to environments where the execution of unauthorized consequence constitutes an irreversible system failure.

Chapter 1: Passive Governance Cannot Own Consequence

Traditional software governance and enterprise security models operate on an architectural fallacy: the belief that inspecting an action after it has been initiated, or alongside its execution path, constitutes control. This structural decoupling between evaluation and realization exposes a fatal systemic vulnerability.

When an architecture separates the path of policy computation from the path of state mutation, it introduces **control-path latency**. In high-throughput, distributed systems, this latency guarantees that the state of the system can alter between the moment a policy decision is rendered and the moment the resulting state change occurs—a failure state defined as **Concurrency-induced State Drift**.

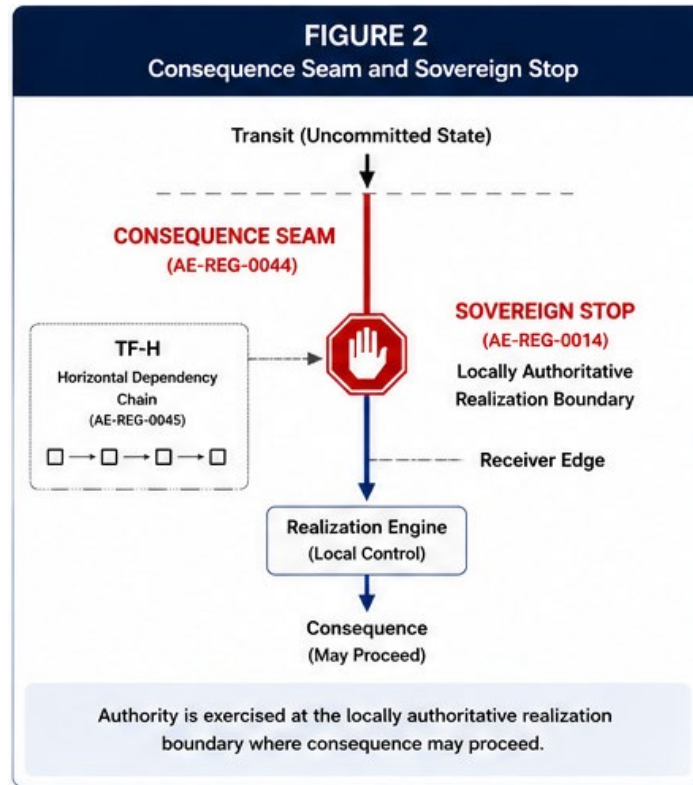


Because passive governance systems sit adjacent to the actual execution plane—relying on asynchronous log aggregation, sidecar introspection, or polling APIs—they are intrinsically retrospective. They report on what *has occurred* rather than regulating what *is occurring*. Consequently, advisory systems do not govern consequence; they merely document its fallout.

When a system cannot guarantee that its evaluation matches the instantaneous execution context at the exact millisecond of state mutation, it yields its sovereignty. Governance separated from realization cannot maintain authoritative regulation of consequence. A governing mechanism must ensure that its evaluation remains structurally congruent with the instantaneous execution state at the exact microsecond of mutation to reliably own consequence.

Chapter 2: Consequence Is Governed Locally

To prevent unauthorized state changes, authority must move from centralized, out-of-band clearinghouses to the exact boundary where data transitions from transit to execution: the **Consequence Seam**. The seam is the precise structural interface where an uncommitted state transition becomes an irreversible operational reality.



Authority Engineering requires governance authority to terminate at the consequence seam through a Sovereign Stop. This architecture mandates **Receiver-edge ownership**, establishing that the entity experiencing the consequence of an action must possess the autonomous capability to terminate the execution path.

Centralizing consequence ownership introduces dependency chains that inherently break down under adversarial conditions or network partitioning. If a local component must query a remote authority to determine whether to halt an unsafe state transition, the local component lacks **Local Stop Authority**. If the communication fabric fails, degrades, or is bypassed, the system defaults to either operational paralysis or un-governed execution. Therefore, authority exists where consequence becomes irreversible.

Chapter 3: Bounded Local Evaluation

If authority must be exercised at the consequence seam, it follows that the inputs required to compute an authority decision must be fully present at that seam. This is the **Local Realization Constraint**.

Bounded Local Evaluation dictates that the enforcement mechanism at the consequence seam must resolve its structural validation using only:

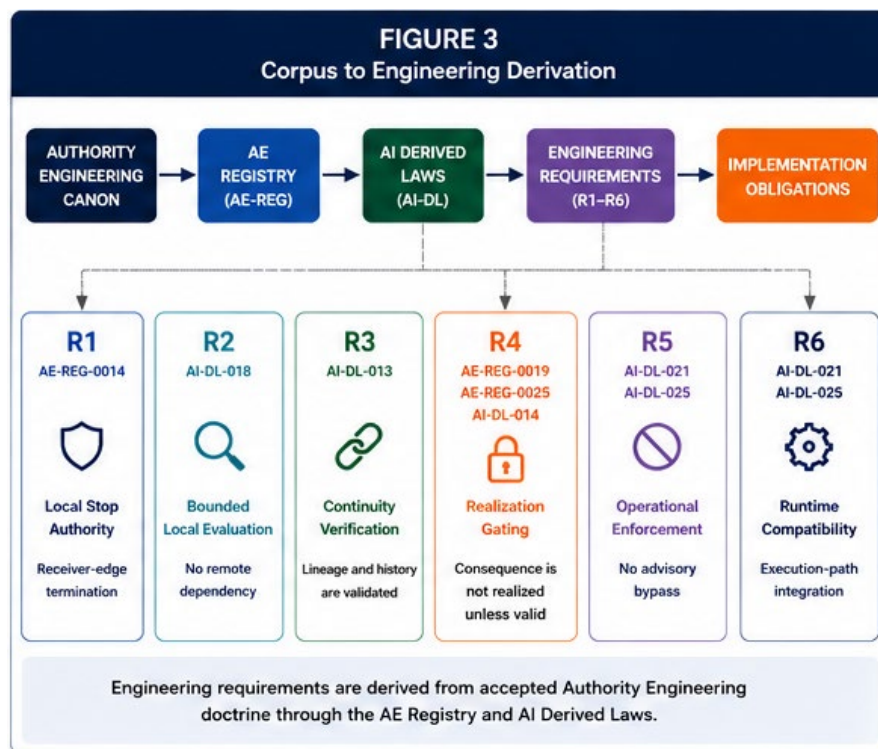
- The immediate, inline context of the arriving payload.
- The continuity history associated with the execution vector.

Relying on out-of-band state reconstruction—such as querying a distributed database or a remote telemetry cache during the execution path—violates the local realization constraint. Reconstruction introduces external dependencies, expands the attack surface, and re-introduces the exact control-path latency that invalidates passive governance.

To maintain **consequence fidelity**—ensuring the actual state change matches the authorized intent—the enforcement mechanism must validate the unbroken lineage of the request locally. Valid governance requires bounded seam-local evaluation.

Chapter 4: Engineering Requirements Derived from Authority Engineering

To bridge the governing constraints established by the canonical corpus to a practical implementation, we translate doctrine into an explicit, immutable engineering specification. Figure 3 serves as the definitive structural contract between Part I theory and Part II realization.



- **R1: Preserve Local Stop Authority:** The architecture must ensure the receiver-edge retains autonomous termination capabilities over state mutations.
- **R2: Perform Bounded Local Evaluation:** The system must evaluate compliance natively at the seam without out-of-band or remote dependencies.
- **R3: Preserve Continuity History:** The system must preserve and validate continuity history sufficient to maintain consequence fidelity.

- **R4: Preserve Consequence Fidelity:** The enforcement mechanism must preserve consistency between realized state mutations and the authorized topology model.
- **R5: Maintain Operational Enforcement:** The governance mechanism must be physically incapable of being placed in a passive, advisory, or bypassed configuration during active execution.
- **R6: Remain Temporally Compatible with Execution:** The processing overhead of the structural checks must operate within the native performance tolerances of the execution plane.

Any implementation claiming compliance with Authority Engineering must satisfy these structural requirements. The remainder of this report presents one implementation satisfying these derived requirements.

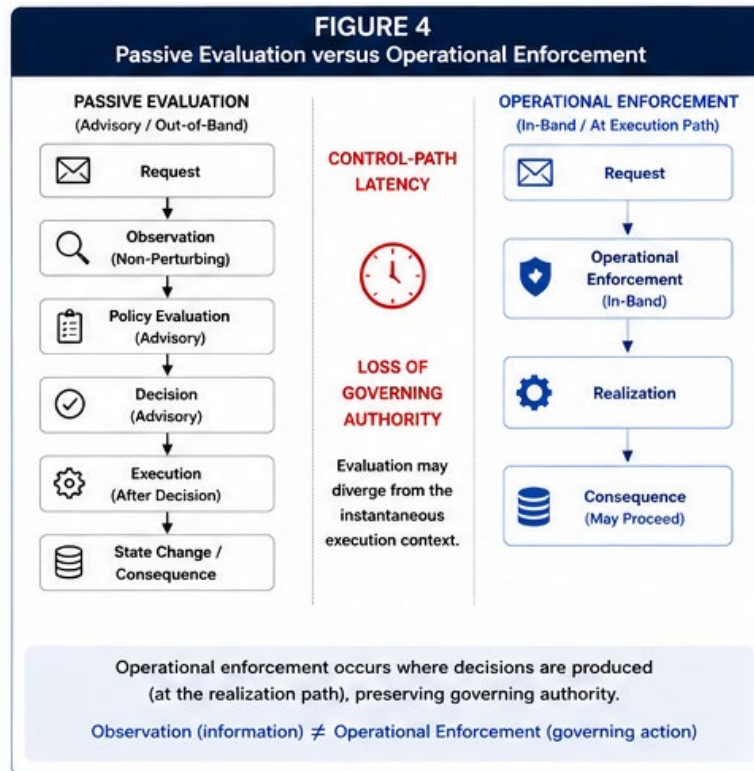
PART II: ENGINEERING REALIZATION

Mandatory Isolation Disclaimer

The implementation presented below is an engineering realization of the requirements established in Part I. It introduces no new Authority Engineering primitives, governing laws, or canonical ontology.

Chapter 5: Passive Evaluation Architectures

Contemporary enterprise access controls uniformly fail the requirements established in Part I. Traditional implementations rely on a decoupled architecture consisting of a Policy Administration Point (PAP), a Policy Decision Point (PDP), and a Policy Enforcement Point (PEP). Whether deployed as an API authorization gateway, an orchestration admission controller, or a network service mesh sidecar, the structural topography remains identical:



In this legacy configuration, the PEP is not a sovereign gate; it is a telemetry collector and a passive switch. The actual evaluation occurs at the PDP, which is frequently located across a network boundary or running as a separate user-space process. Because these architectures rely on text-based policy engines or dynamic lookups during the request lifecycle, they cannot natively bind the evaluation to the physical state mutation. They function purely as advisory systems, providing a window of vulnerability where execution can deviate wildly from authority before an alert is generated or an out-of-band block is issued.

Chapter 6: Real-World Failure Patterns

The following examples are presented as representative architectural patterns rather than exhaustive evaluations of specific products or implementations. They illustrate how the structural properties derived in Part I appear in widely deployed enterprise systems.

Case 1: Time-of-Check to Time-of-Use (TOCTOU) in Orchestration Engines

In cluster orchestration topologies, Validating and Mutating Admission Controllers serve as out-of-band Policy Decision Points. When an execution request arrives, the API gateway delegates validation to these controllers. Once approved, the configuration state is committed to a distributed data store. However, the actual resource execution occurs downstream at the worker node level at a later time.

The architecture introduces a temporal separation between admission-time evaluation and node-level realization. Between the time of validation at the cluster control plane and the time of realization at the node level, the structural context can alter, leaving the local execution node entirely blind to the original validation context and incapable of enforcing inline invariants at the point of realization.

Case 2: Structural Decoupling in General-Purpose Policy Engines

General-purpose policy frameworks frequently function as out-of-band evaluation sidecars or localized API daemons. The host application forwards a structured request context to the engine, receives a Boolean response, and is then expected to enforce that decision.

Because enforcement remains the responsibility of the host application, correctness depends on faithful integration between evaluation and realization. The policy engine itself possesses zero native realization authority. Because the evaluation mechanism does not sit directly on the execution seam, the underlying application logic may execute out of alignment with the target decision under extreme load or resource pressure.

Case 3: Distributed State Fragment in Hyperscale Identity Provider (IdP) Fabrics

Hyperscale Cloud Identity and Access Management (IAM) environments operate with a centralized policy administration and decision layer. When an execution command traverses the cloud fabric, the local service edge requests authorization from a remote IAM evaluation service.

Because the central policy engine does not own realization within the hundreds of underlying, highly distributed cloud services, the local service edge must rely on a cached copy or an asynchronous evaluation loop. Under network partitioning conditions, this design forces an architectural dilemma: the local service edge must either drop the traffic (inducing operational paralysis) or process the state transition based on potentially stale data, violating the requirement for consequence fidelity.

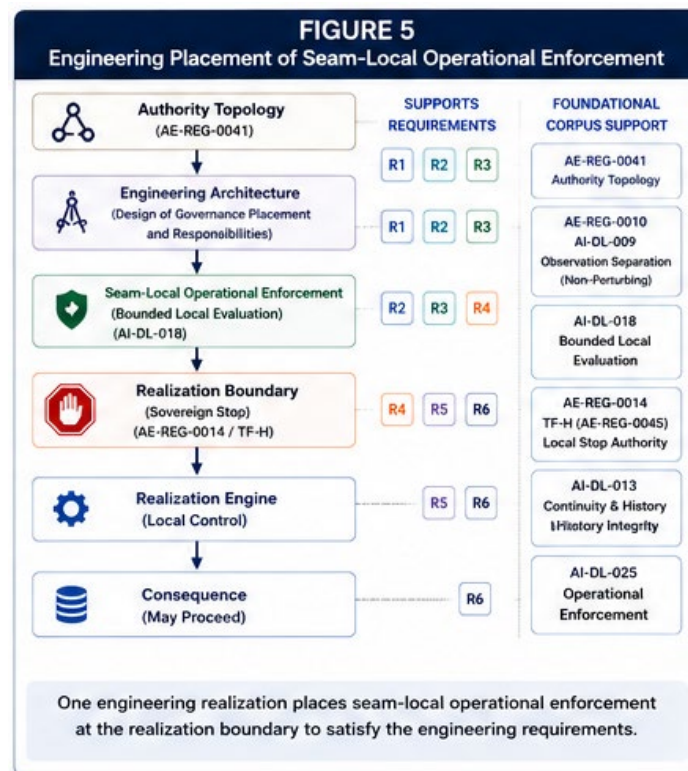
Case 4: The Reactive Failure Mode in Enterprise Ransomware Proliferation

The operational danger of retrospective governance is illustrated by standard enterprise cyber defense topologies. Classic Endpoint Detection and Response (EDR) agents monitor filesystem state changes via asynchronous kernel callbacks or log forwarding.

The execution loop flows from detection to telemetry ingestion, alert generation, centralized analysis engine verification, and automated containment response. By the time this decoupled, advisory loop renders a containment decision, the unauthorized consequence—irreversible block-level cryptographic encryption of the filesystem—has already occurred. The governance model is unable to preserve consequence fidelity because it lacks inline, seam-local structural constraints capable of executing a sovereign stop prior to data write.

Chapter 7: Seam-Local Structural Enforcement

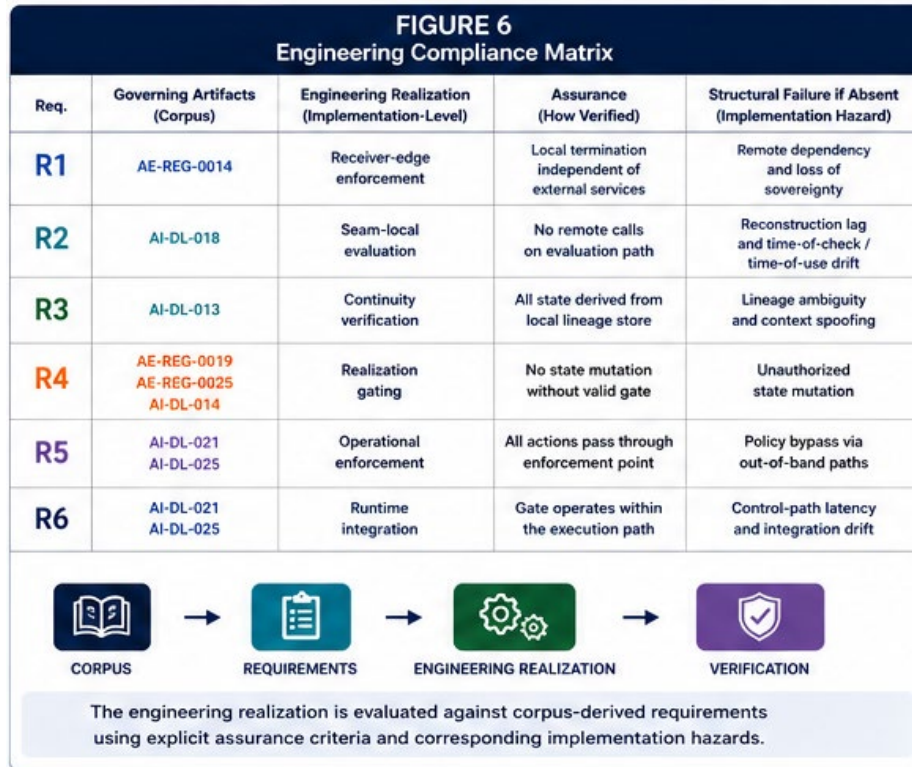
To fulfill the derived engineering specification, one realization architecture eliminates the decoupled policy evaluation loop entirely. The paradigm shifts from a reactive model to an immutable, inline structural model:



Under this framework, the validation mechanism is embedded into the data plane at the consequence seam. The request cannot be parsed, scheduled, or allocated resources by the realization engine unless it physically passes through an inline constraint layer. This layer does not evaluate policy strings or calculate dynamic permissions at runtime; instead, it enforces the predefined, valid structural boundaries of the data transit path itself.

Chapter 8: Translating Authority into Execution Constraints

One implementation satisfying the engineering requirements translates the governing authority topology model into executable structural constraints. Rather than interpreting raw authority models during the critical execution path, the governing topology model is processed beforehand into optimized, deployment-ready structures.



During deployment, these constraints are compiled into optimized runtime structures, comprising deterministic state tables and cryptographic lineage verifications to satisfy the requirement for continuity history (R3). When deployed directly to the runtime environment, these constraints act as an immutable structural enforcement layer. The runtime data plane reads the incoming payload context, checks it against the localized state table, and instantly permits or terminates the path based purely on structural alignment.

Chapter 9: The Deterministic Data Plane

The operational manifestation of this architecture is the **Deterministic Data Plane**. By satisfying the corpus-derived engineering requirements established in Part I, this implementation yields several emergent operational properties that fundamentally separate it from legacy access frameworks:

- **Elimination of Retrospective Governance:** By collapsing evaluation directly into the execution path, the system removes the ability to merely document post-execution failures, replacing tracking with structural boundary confinement.
- **Elimination of Secondary Policy Decision Paths:** In this implementation, secondary policy decision paths, advisory modes, and fallback decision states are intentionally

eliminated from the governed execution flow. The enforcement path *is* the execution path.

- **Execution-Time Authority Enforcement:** Requests lacking a structurally valid execution path are terminated instantly at the receiver edge prior to resource invocation or state initialization.
- **Receiver-Edge Consequence Ownership:** Local nodes maintain autonomous, terminal control over arriving vectors, rendering them resilient against out-of-band communication partition failures.
- **Local Consequence Regulation:** The system bounds all evaluation criteria to localized state tables and inline context payloads, minimizing the shared runtime attack surface.
- **Preservation of Consequence Fidelity:** Under the assumed operating conditions established in Part I, the data plane ensures that no mutation occurs outside the boundary of the authorized structural state.

Realization is thus rendered entirely deterministic; mutations are blocked unless they conform strictly to the structural boundaries established by the authority model.

Chapter 10: Operational Realization Tradeoffs

Transitioning to an inline structural enforcement layer demands rigorous engineering compromises. The benefits of consequence fidelity are counterbalanced by specific system costs that must be managed by the implementation architecture:

- **Offline Preprocessing Overhead:** Compiling large, enterprise-wide authority topology models into low-level executable constraint sets requires significant up-front computational processing. While this removes policy-parsing delays from the hot execution path, it increases the initial assembly and validation times within continuous delivery pipelines.
- **Deployment Complexity and Initialization States:** Because every edge node must operate with full structural autonomy, the system requires highly reliable initialization pathways to safely load binary state tables before exposing data-plane interfaces. An uninitialized or partially loaded node must default to a hard, operational lockdown, prioritizing security over system availability during local restarts. Initialization vectors must rely on locally immutable cryptographic seeds embedded at compile-time to verify the first incoming state-table transit path.
- **Update Cadence and State Synchronization Propagation:** When the governing authority topology model is updated, the changes must be re-compiled and pushed to every distributed edge node. Under high-churn topology environments, managing the asynchronous propagation of these pre-compiled immutable layers can lead to short windows where different edges are executing minor variances of the constraint set, requiring robust transaction boundaries to prevent state discrepancies.

- **Edge Recovery and Rollback Coordination:** In the event of an invalid configuration deployment or a corrupted state propagation, edge recovery must be managed without introducing out-of-band coordination channels that bypass local constraints. Rolling updates and rollbacks require a strict, locally verifiable lineage progression, ensuring that nodes can autonomously revert to a previous, cryptographically verified structural state table without losing Local Stop Authority during the transition window.
- **Increased Memory Footprint at the Edge:** Storing complete, multi-dimensional state tables and historical lineage metrics locally on the receiver edge increases local memory consumption. This overhead necessitates dense, memory-mapped binary layouts (e.g., zero-copy serialization frameworks) to ensure deterministic lookups on resource-constrained edge hardware.
- **Rigid Execution Compatibility Requirements:** Because the data plane acts as an absolute structural filter, legacy applications or uncoordinated traffic flows that deviate even slightly from the authorized schema will be immediately dropped. This necessitates strict schema discipline and exhaustive contract mapping prior to activation.

Chapter 11: Scientific Falsification Framework

To maintain scientific validity and empirical rigor, this technical report establishes a dual-layered falsification framework. The validity of this document's conclusions is contingent upon the non-existence of architectural states that violate these explicit boundary conditions.

Theory Falsification (Attacks on Authority Engineering Premises)

The theoretical claims developed in Part I would be materially challenged if any of the following boundary conditions are empirically demonstrated in a production environment:

- **T1:** An out-of-band, centralized policy database maintains a node's Local Stop Authority (R1) during a complete network partition without inducing node paralysis.
- **T2:** A retrospective governance architecture consistently prevents unauthorized state mutations at the consequence seam under arbitrary concurrent execution loads.
- **T3:** A distributed execution system maintains absolute consequence fidelity (R4) under arbitrary mutation using remote state reconstruction that operates entirely outside the local realization path.

Implementation Falsification (Attacks on the Deterministic Data Plane Realization)

The concrete realization claims established in Part II would be materially challenged if any of the following mechanical properties are empirically observed:

- **I1:** An inline data-plane proxy implementation satisfies the requirement for Operational Enforcement (R5) while simultaneously supporting a dynamic bypass or alert-only configuration option.

- **I2:** The pre-compiled executable constraint sets fail to accurately map the structural bounds of the original authority topology model, allowing unexpected, out-of-bounds execution paths to materialize.
- **I3:** The computational latency overhead (Δt) induced by validating localized state tables and cryptographic lineage scales non-linearly with the depth of the continuity history, exceeding the sub-millisecond tolerances required by high-throughput runtimes.
- **I4:** An alternative engineering realization successfully satisfies the full specification contract (R1–R6) under active partition and concurrent production mutation without utilizing a seam-local enforcement architecture.

Chapter 12: Compliance with Authority Engineering

To verify that this specific realization satisfies the structural criteria established in Part I, we map the concrete operational mechanisms of the Deterministic Data Plane directly back to the derived engineering requirements, noting the precise failure domain prevented by each design:

Derived Requirement	Engineering Mechanism	Verification Metric / Operational State	Failure if Absent
R1: Local Stop Authority	Receiver-Edge Enforcement	Data plane terminates execution path <i>prior</i> to consequence realization if validation fails.	Remote Dependency: Downward node failure paths override governance.
R2: Bounded Local Evaluation	Seam-Local Evaluation	Validation relies entirely on localized state tables and inline context; zero network lookups.	Reconstruction Lag: Latency loops open windows for race-condition execution errors.
R3: Continuity History	Cryptographic Lineage Verification	Cryptographic lineage and ancestry tokens are verified inline at the packet/payload layer.	Lineage Ambiguity: Unverified ancestral paths allow unauthorized structural injection.

Derived Requirement	Engineering Mechanism	Verification Metric / Operational State	Failure if Absent
R4: Consequence Fidelity	Realization Gating	State mutations are structurally blocked unless they map identically to the pre-compiled topology.	Unauthorized Mutation: State drift yields divergence from authorized blueprints.
R5: Operational Enforcement	Inline Structural Enforcement	Enforcement gate is structurally inseparable from the network proxy data plane fabric.	Policy Bypass: Misconfigurations allow administrative or advisory out-of-band escape.
R6: Temporal Compatibility	Data-Plane Integration	Memory-mapped state tables ensure constraint validation executes within sub-millisecond latencies.	Control-Path Lag: Latency overruns prompt performance drops or timeout bypasses.

Figure 7: Traceability and Compliance Matrix.

Conclusion

Authority Engineering establishes the invariant governing constraints that must regulate any system where state mutation bears operational consequence. As derived in Part I of this report, traditional governance models are fundamentally incapable of honoring these constraints due to their architectural reliance on passive, out-of-band, and retrospective evaluation loops.

This technical report has presented one engineering realization that successfully translates canonical authority doctrine into concrete execution-time mechanics. By compiling authority topologies directly into inline structural constraints, the Deterministic Data Plane is designed to preserve consequence fidelity under the governing assumptions established in Part I—all without introducing new primitives or modifying the foundational canon of Authority Engineering. Future implementations may differ significantly in mechanism while remaining fully compliant with these identical governing requirements.