DPI GOVERNANCE

DPI: Execution Governance for AI Systems

How declared authority, scope, and refusal become enforceable before action

> Governance becomes real when extra-mandate action is structurally impossible, not just traceable.

White Paper · Draft v0.5
Nemo Flow Systems · DPI Governance

A public framing of execution governance centered on the allow/deny boundary, runtime refusal, and verifiable receipts.

Executive Summary

Most AI governance today operates outside the system.

It appears in policy documents, model cards, risk taxonomies, audit controls, oversight committees, incident reports, and post-hoc evaluations. These structures matter. They improve visibility, accountability, and institutional preparedness. But they do not fully solve the deepest operational problem: consequential action still occurs at execution time.

If an AI system can act beyond its mandate when the moment of execution arrives, governance remains incomplete.

This paper introduces DPI as an execution-governance architecture centered on the allow/deny boundary: the runtime point at which an action is either permitted within declared authority and scope, or refused before consequence.

DPI is not designed to explain systems after the fact. It is designed to constrain what action is allowed to exist in the first place.

1. The Problem: Governance Outside the System

Organizations now invest in model documentation, safety evaluations, risk classifications, red-team exercises, review boards, audit processes, human-in-the-loop procedures, and policy workflows. These mechanisms improve awareness and accountability.

But a critical gap remains. Most governance still lives outside the system. It exists in documents, policies, oversight channels, and reporting structures while the system continues toward execution.

The moment that matters most is not the moment of documentation. It is the moment an action is attempted.

A system may be extensively documented and still inadequately governed. It may have robust oversight and still retain the operational ability to act beyond intended scope.

2. Why Existing Approaches Are Incomplete

Current governance methods cluster into descriptive governance, evaluative governance, supervisory governance, reactive governance, and procedural governance. Each of these plays a role. None should be dismissed.

However, they share a common limitation: they do not necessarily determine whether a specific action may become real at runtime.

A system can be documented without being constrained, monitored without being refused, auditable without being bounded, and explainable without being governable at execution.

Traceability is useful, but it is not enough. A trace can tell us what happened; it cannot by itself guarantee that the event should have been impossible.

3. Execution Governance as the Missing Layer

Execution governance begins from a narrower question than most AI governance discourse asks.

Not: Was the system described properly? Was the risk categorized? Was the model aligned in general terms?

But: Can this action happen right now? Under whose authority? Within what mandate? Under which constraints? If those conditions fail, can the system refuse?

Execution governance exists at the point where state changes, consequences unfold, and reality is affected. Its purpose is to determine whether execution is legitimate in the moment before action becomes real.

4. DPI: Core Model

DPI is an execution-governance architecture centered on the allow/deny boundary.

At this boundary, an attempted action is evaluated against declared purpose, explicit authority, mandate scope, runtime refusal, and receipt generation.

The boundary resolves to one of two outcomes: ALLOW or DENY. This binary matters because it forces governance to become operational rather than aspirational.

DPI does not treat refusal as an exception to governance. It treats refusal as one of governance's core functions.

5. The Execution Boundary

The execution boundary is the point at which an attempted action meets a governing structure that can admit or refuse it.

It is not a dashboard, a policy memo, an ethics statement, or a post-event report. It is a live, runtime decision point.

A real execution boundary has four properties: pre-execution position, binding logic, refusal capability, and legible evidence.

6. Structural Impossibility Outside Mandate

The central design principle of DPI is that governance is incomplete if extra-mandate action remains operationally possible.

This does not mean all failure can be eliminated. It means the architecture should be designed such that action outside valid mandate is not merely discouraged, flagged, or logged. It should be blocked at the boundary as a matter of execution structure.

If the system lacks valid authority, declared purpose alignment, or scope compatibility, it does not act. This is where governance becomes real.

7. Receipts, Evidence, and Verification

Execution governance requires more than refusal. It also requires legibility.

Every allow or deny outcome should produce a receipt that records the decision in a way that is append-only, reviewable, verifiable, and bound to governing context.

Reports describe. Receipts prove.

A DPI-style receipt should preserve enough information to reconstruct the boundary judgment without exposing unnecessary internal logic.

8. DPI vs. Policy Engines and Audit Systems

DPI overlaps with several existing control categories, but it is not identical to them.

Policy engines evaluate rules and determine whether requests satisfy configured policies. Audit systems reconstruct what happened. Monitoring observes and flags. Access control restricts resources. Human review adds a checkpoint.

The simplest contrast is this: monitoring sees drift, audit records drift, policy describes permitted drift, DPI refuses the transition.

9. Governed Commit Path

A useful way to understand DPI is through the idea of a governed commit path.

Many systems can generate outputs, recommendations, plans, or candidate actions. But not every output changes reality. What matters is the path by which a suggestion becomes a committed action.

Without such a path, systems may be surrounded by governance while still retaining a route to illegitimate action. DPI is concerned with the narrow point where one of many possible actions becomes real.

10. Practical Implementation Path

DPI does not need to begin as a full operating system or universal standard. It can begin as a bounded runtime layer implemented around high-consequence actions.

A practical implementation path includes defining declared purpose, defining mandate scope, binding authority, intercepting attempted actions, refusing out-of-scope transitions, emitting receipts, and enabling independent verification.

The important point is not scale first. It is legitimacy first.

11. Example Use Cases

DPI is most useful where system action carries meaningful consequence: financial action, infrastructure action, data action, and high-risk research or lab

action.

In each case, the same question applies: Can the system act beyond mandate? If yes, governance is still incomplete.

12. What DPI Is Not

DPI is not a general AI philosophy, not a chatbot, not a generic compliance product, not a trust-and-safety branding layer, and not a complete theory of intelligence.

It is an execution-governance architecture focused on the runtime boundary where action is either admitted or refused.

13. Limits and Open Questions

DPI is not a claim of total safety. Hard questions remain around sufficiency of context, authority revocation, public versus private evidence, degraded-mode behavior, chained actions, and multi-system interaction.

DPI does not eliminate these questions. It creates a structure in which they can be asked more precisely.

14. Conclusion

The future of AI governance will not be determined by explanation alone. It will be determined by whether systems can be kept inside mandate when action becomes real.

The central question is no longer: Did we document the risk? It is: Can the system act beyond its mandate?

If the answer is yes, governance is still incomplete. DPI proposes a different standard: a runtime boundary where authority, purpose, scope, and refusal are enforced before consequence, and where extra-mandate action becomes structurally impossible rather than merely traceable.

Figures and Comparative Tables

The figures and tables below illustrate the central claim of this paper: most AI governance mechanisms improve visibility around the system, but do not yet determine whether a consequential action may become real at runtime. DPI addresses this gap by relocating governance to the execution boundary, where declared purpose, authority, scope, and refusal are evaluated before consequence.

Figure 1. The Governance Gap

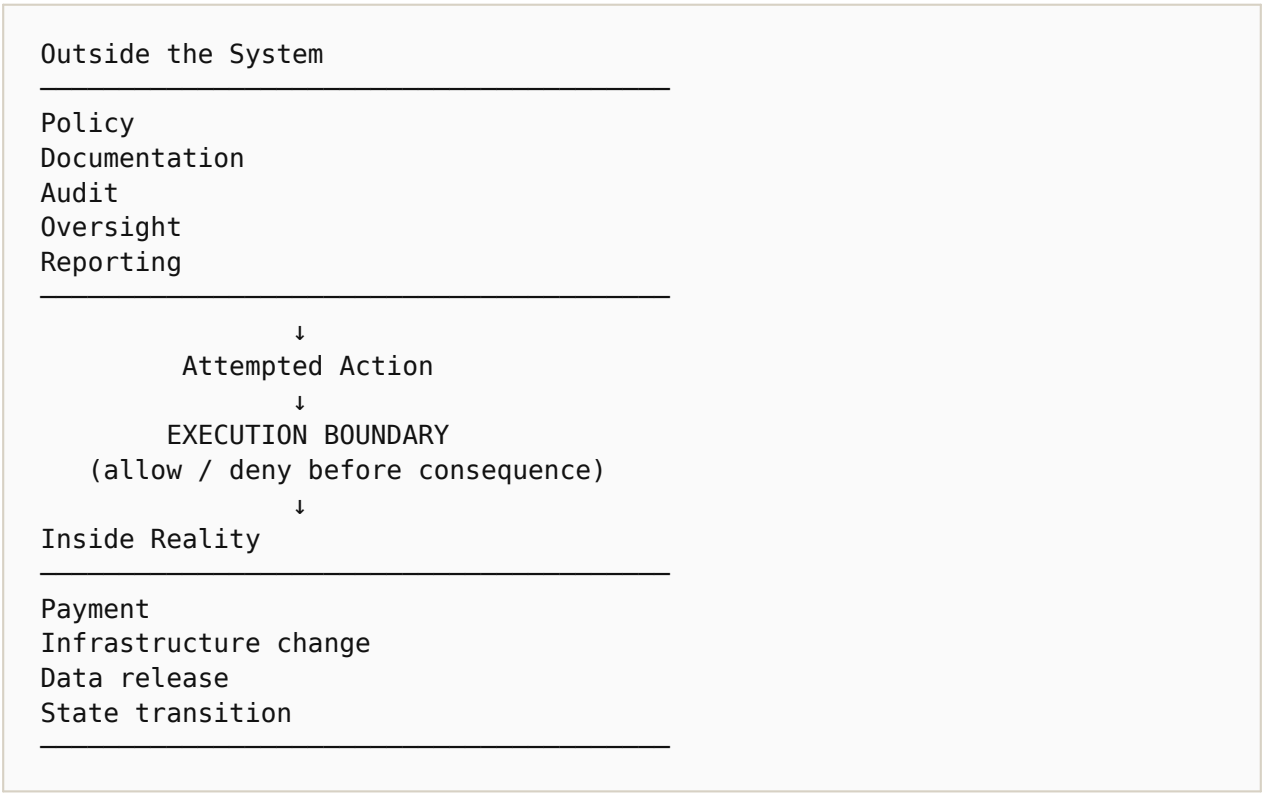

Figure 1. The governance gap emerges when oversight remains outside the system while real consequence is produced at execution.

Figure 2. DPI Execution Boundary

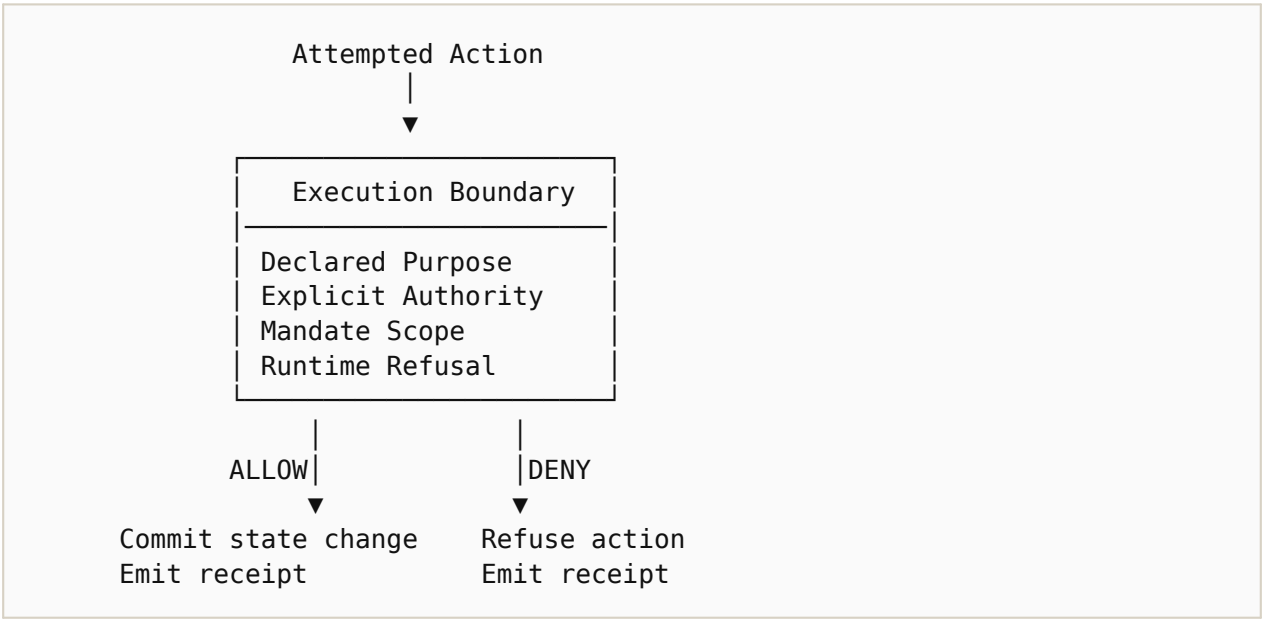

Figure 2. DPI positions governance at the runtime boundary where action is either admitted within mandate or refused before consequence.

Figure 3. From Traceability to Structural Impossibility

```
Old Model: Action → Harm / Drift → Log → Audit → Explanation

DPI Model: Action Attempt → Boundary Check → Allow or Deny → Receipt
```

Figure 3. The shift from traceability to structural impossibility marks the transition from governance as commentary to governance as real execution control.

Table 1. DPI Compared with Existing Governance Approaches

Approach	When	Produces	Core Limitation
Policy documentation	Before	Rules, statements	Does not bind action at runtime
Audit systems	After	Reports, findings	Explains violation after consequence
Monitoring systems	During/After	Alerts, telemetry	Sees problems but may not stop them
Policy engines	Before/During	Permit / deny	Often rule-centric, not mandate-centric
Human-in-the-loop	Before	Approval / rejection	Weak if authority or scope are unclear
Access control	Before	Auth outcomes	Identity alone does not govern consequential action
DPI / Execution Governance	At execution	Decision + receipt	Requires tight authority / scope definition

Table 2. Commentary vs. Real Governance

Test	Commentary	Execution Governance
Can it describe the risk?	Yes	Yes
Can it explain what happened?	Yes	Yes
Can it refuse action at execution?	Not necessarily	Yes
Can it prevent extra-mandate action from becoming real?	Not necessarily	Yes
Does it produce proof of allow/deny?	Sometimes indirectly	Yes, directly
Is violation traceable or structurally impossible?	Usually traceable	Structurally impossible outside mandate

Table 3. Soft Control vs. Binding Control

Dimension	Soft / Non-Binding Control	Binding / DPI-Style Control
Authority	Inferred	Explicitly bound
Action	Suggested	Admitted or refused
Scope	Advisory	Enforceable
Consequence	May still proceed	Blocked outside mandate
Evidence	Report	Receipt
Governance location	Around the system	At the execution boundary

Appendix: Short Public Definitions

Execution governance — The runtime governance layer that determines whether a system action may proceed or must be refused before consequence.

Execution boundary — The allow/deny point where declared purpose, authority, and scope are evaluated against an attempted action.

Structural impossibility outside mandate — A condition in which extra-mandate action cannot become real because the boundary blocks the transition itself.

Receipt — A verifiable record that an action was evaluated at the boundary and was either allowed or denied under the governing context.