# Governance Intake

*The Missing First Step in Runtime AI Governance*

*PV-PP Framework White Paper v0.1 | Amundsen R&D LLC | May 2026*

## Executive Summary

Runtime AI governance is often described as a control problem at the moment of execution: before an agent sends a message, moves a workflow, triggers a payment, routes a case, or invokes a tool, the system must decide whether the proposed action is admissible.

That framing is correct, but incomplete. A runtime sidecar cannot enforce governance that has not first been extracted, normalized, and tested. Real organizations rarely keep governance answers in one clean place. The rules are scattered across policies, SOPs, forms, workflow screenshots, audit findings, training materials, exception procedures, sample tickets, and informal practice.

This paper argues that runtime AI governance needs a front-end requirements layer: governance intake. Governance intake converts messy organizational materials into a source-ranked extraction brief, identifies conflicts and gaps, asks targeted questions, and only then compiles a sidecar-ready packet. Without that step, governance builders collapse into long questionnaires that real users cannot answer reliably.

> ***Central claim:*** *Runtime governance starts before runtime. The enforceable control point depends on an upstream intake process that discovers the organization's actual rules before those rules can be compiled into executable form.*

## 1. The Questionnaire Failure Mode

A common mistake in AI governance tooling is to assume that a user can answer all setup questions directly. What actions can the system take? Who has authority? What evidence is required? What blocks execution? What should escalate? Which thresholds govern? Which documents outrank others? For a demo, those answers can be invented. For a real organization, they usually cannot.

The person operating a governance builder may know the workflow, but not every policy exception, source hierarchy, audit constraint, or downstream failure mode. Even when the information exists, it is often dispersed across departments and document types. A questionnaire-first workflow therefore creates a false precision problem: the user is pushed to answer from memory, and the resulting packet inherits those gaps.

This is not a minor UX issue. It is a governance-design defect. A runtime sidecar can only enforce the conditions it is given. If the organization's rules have not been discovered and normalized, then the sidecar either blocks too much, allows too much, or quietly encodes the wrong assumptions.

## 2. Governance Intake as the Missing Layer

Governance intake is the process of turning disjoint organizational artifacts into a structured governance model. It should precede ordinary discovery and packet compilation. The user should not begin by answering a forty-page questionnaire. The user should begin by uploading what exists: policies, SOPs, forms, screenshots, escalation matrices, sample cases, audit findings, system field definitions, exception procedures, and existing agent prompts or workflow logic.

The intake layer should then extract candidate facts, not final truth. It should separate what is known from source material, what is inferred but needs confirmation, what is missing, what is contradictory, and what belongs outside the static Phase 1 boundary.

## 3. The Intake-to-Runtime Pipeline

| 1. Intake | 2. Extraction | 3. Resolution | 4. Compilation | 5. Validation |
|---|---|---|---|---|
| Upload first batch of org materials | Source map<br>extraction brief<br>gap report | Targeted questions<br>conflict resolution | Strict sidecar packet<br>field/operator/value rules | Test cases<br>runner<br>PASS/FAIL result |

The important shift is that the governance assistant asks broad questions only after document extraction. The system can say, for example, “Policy A gives a $75 receipt threshold, while a training FAQ says $50. Policy A appears higher authority. Confirm which threshold governs.” That is materially different from asking the user to reconstruct the threshold from memory.

## 4. Source Authority Is Part of Governance

Document intake is not simple summarization. The system must preserve source authority. Current official policy should normally outrank training slides. Required fields in a system form may reveal operational constraints not written in a policy. Audit findings may expose failure modes absent from happy-path procedures. Informal practice may explain how people work, but it should not silently override an official control.

A governance intake system should therefore produce a source map. Each source should be identified, ranked, and linked to extracted claims. Conflicts should not be merged into a synthetic answer. They should be surfaced as decisions requiring confirmation.

| Approximate Rank | Source Type |
|---|---|
| 1 | Official policy, legal, compliance, or regulatory document |
| 2 | Current SOP or process manual |
| 3 | System form or required-field specification |
| 4 | Escalation matrix or authority table |
| 5 | Training materials |
| 6 | Sample cases or tickets |
| 7 | Informal email, chat, or tribal practice |
| 8 | User explanation |
| 9 | Model inference |

## 5. From Governance Language to Executable Rules

After extraction and gap resolution, the English model has to be compiled. A sidecar packet is not a policy essay. It must define explicit fields, enums, operators, decisions, precedence, and trace outputs.

Phrases such as “verify authority,” “check compliance,” or “make sure it is safe” are not executable governance rules. They have to become declared packet fields and predicate rules with pass values, failure decisions, and trace fields.

| Vague Governance Language | Executable Sidecar Form |
|---|---|
| Check authority | approval_authority_present == false -> REQUEST_MORE_INFORMATION |
| Verify documents | proof_of_purchase_present == false -> REQUEST_MORE_INFORMATION |
| Assess safety risk | safety_issue_reported == true -> ESCALATE_HUMAN_REVIEW |

| | |
|---|---|
| Prevent fraud | fabricated_evidence_suspected == true -> BLOCK_AUTOMATED_RESOLUTION |

## 6. Testing Is Governance, Not Cleanup

A compiled packet is still not complete until it is tested. Tests should include ordinary approvals, blocks, escalations, information requests, invalid schemas, edge thresholds, precedence conflicts, and scalar-trap cases where an optimization signal tries to override an admissibility failure.

A runner should load the packet, load test cases, evaluate predicates, apply precedence, and print pass/fail results. If the packet and tests disagree, that is not just a coding issue. It may reveal an unresolved governance contradiction, a stale test assumption, or a compiled rule that did not match the confirmed model.

The completion condition should be explicit: the scenario is not complete until the test suite passes, or any unresolved failures are identified and accepted as open work.

## 7. PV-PP Framing

In the PV-PP framework, the relevant question is not merely what an agent or tool can do. The question is whether a proposed action is viable under the governing domains that preserve authority, evidence, adequacy, recovery, constraints, and traceability.

Governance intake supports that ordering. It prevents a governance builder from jumping directly to optimization or action selection before the organization's admissibility conditions have been identified. It also keeps Phase 1 static governance distinct from Phase 2 live runtime checks. If a rule requires a live lookup, live OCR, live image analysis, current inventory, or account-state refresh, it should either be represented as a declared packet field or classified as a later runtime capability.

## 8. Conclusion

The practical lesson is simple: runtime governance starts before runtime.

The first operational problem is not only how to enforce a rule at the execution boundary. It is how to discover the rule, rank its source, resolve conflicts, normalize it into packet fields, and test it before it is allowed to govern anything real.

A questionnaire-first governance tool will work for demos and controlled prototypes. It will fail in organizations where rules are scattered, contradictory, and partly informal. A document-first governance intake layer changes the workflow. It lets an organization start with what it actually has and move toward executable control one verified step at a time.

A sidecar cannot enforce governance that has never been extracted into executable form. Governance intake is the missing first step.

**Suggested LinkedIn reference line:** "A sidecar cannot enforce governance that has never been extracted into executable form."