PV-PP Agent Governance White Paper

Viability Control for AI Agents, Tools, and Runtime Decisions

Version 0.1 | May 2026 | External-Facing Draft

Positioning note: This white paper describes the PV-PP framework as an agent-governance layer. It is not a certification claim, legal opinion, safety guarantee, or proof that a given deployed system is safe. It is a product and architecture framing document.

1. Executive Summary

AI agents are moving from chat interfaces into operational systems. They retrieve data, call tools, route work, write records, classify risk, trigger workflows, and sometimes create practical authority even when a human remains nominally in charge. The central governance problem is no longer only whether an agent can produce an answer. The problem is whether the agent should be allowed to act, route, escalate, finalize, or rely on a source under the current conditions.

The PV-PP framework offers a governance layer for that problem. It distinguishes what is available from what is viable. A tool may be available through an API, an agent may be capable of invoking it, and a model may be confident about the next step. None of that is sufficient. The relevant question is whether the proposed action preserves the governing corridors required for operational, legal, financial, safety, trust, privacy, reliability, and recovery integrity.

The proposed PV-PP Agent Governance Runtime is a runtime layer sitting above or beside agent orchestrators, tool registries, MCP-style capability layers, workflow engines, and human-review queues. Its function is not to replace those systems. Its function is to decide whether a proposed agent action is viable under the current state, pressure, permissions, evidence, memory, constraints, and recovery conditions.

- Core distinction: MCP-like layers answer “what tools or resources are available?” The PV-PP runtime answers “what actions remain viable?”
- Core product claim: agent governance should be corridor-preserving, not single-score optimizing.
- Core risk claim: an agent can improve visible metrics while damaging hidden governing domains.
- Core implementation claim: viability checks can be expressed as reusable runtime patterns: eligibility gates, source-authority checks, memory/state checks, adequacy gates, escalation corridors, rollback checks, and guarded execution.

2. The Agent Governance Problem

The present agent stack is improving quickly at three layers: models, tools, and orchestration. Models reason better. Tool registries expose more capabilities. Orchestrators coordinate multi-step workflows. But governance remains structurally weaker than capability exposure. Many systems can now identify an available action faster than they can determine whether that action should be admitted.

This creates a predictable failure pattern. An agent appears to succeed because it completes tasks, reduces backlog, closes cases, lowers review volume, or accelerates throughput. Yet the same behavior may shift risk into domains that are not visible on the dashboard: fraud loss, privacy leakage, bad records, broken audit trails, latent compliance exposure, trust decay, or loss of recovery options.

Visible agent success	Hidden governance failure
Faster case closure	Closed before the underlying state is actually resolved
Lower escalation rate	High-risk ambiguous cases silently treated as routine
Higher tool-use efficiency	Dangerous tool combinations become easier to trigger
Reduced manual review	Human review becomes nominal while agent labels become de facto authority
Higher apparent accuracy	Errors concentrate in rare but severe cases that dominate risk
More autonomous workflows	Rollback, appeal, audit, and recovery corridors weaken

This is not a marginal problem. It is structural. Agents are often deployed into environments where local success metrics are easier to measure than downstream viability. Speed, closure rate, and cost reduction are visible. Recoverability, source authority, obligation history, ambiguous evidence, de facto finalization, and regime change are harder to measure. A governance layer must therefore preserve the domains that the agent is not naturally incentivized to protect.

3. PV-PP in Plain Terms

Productive Value (PV) refers to what flows through a system: benefits, harms, services, risks, obligations, resources, permissions, records, trust effects, and operational consequences. Productive Power (PP) refers to the underlying capacity of an agent, organization, user, or system to continue functioning and producing value across relevant domains.

The PV-PP framework treats decisions as bounded comparative governance problems. It does not reduce every outcome to a single utility score. Instead, it asks whether a proposed policy or action preserves the required corridors of viability across governing domains. In agent systems, those domains include but are not limited to data integrity, security, privacy, legal integrity, operational continuity, accountability, recoverability, user trust, and financial integrity.

The practical difference is simple:

- A scalar scoring system may ask which action has the highest expected score.
- A rules engine may ask whether a fixed rule blocks the action.
- A tool router may ask which tool can perform the action.
- The PV-PP governance layer asks whether the action remains viable after pressure, evidence, memory, constraints, permissions, recovery, and downstream domain effects are considered.

4. Available Is Not Viable

The most important product distinction is the gap between availability and viability.

Layer	Primary question	Typical output	Limitation
Tool registry / MCP-like capability layer	What tools, APIs, resources, or functions are available?	List of available tools and schemas	Availability does not imply safe or appropriate use.
Agent orchestrator	How should the agent complete the task?	Plan, tool calls, subagent routing, workflow steps	Planning may optimize task completion without preserving governing domains.
Rules / policy engine	Does a known rule block the action?	Allow, deny, require condition	Rules may miss context, memory, pressure, recovery, and de facto authority.
PV-PP Agent Governance Runtime	Is the proposed action viable under current conditions?	Admit, constrain, escalate, defer, repair, or reject	Requires structured state, source hierarchy, and implementation discipline.

This distinction becomes decisive when an agent has technically limited authority but operationally significant influence. An agent may not release funds, approve a claim, fire an employee, publish a document, or close a legal matter. But if it marks a case "ready," "approved," "low risk," "complete," "routine," or "resolved," downstream humans may rely on that label. The label becomes practical authority. The PV-PP governance layer treats such labels as governing actions, not clerical metadata.

5. Why Existing Controls Are Not Enough

The PV-PP runtime is not a replacement for security, policy, compliance, observability, or ordinary workflow controls. Those controls remain necessary. The issue is that each covers only part of the governance problem.

Control type	What it handles well	What it misses
Access control	Who or what can use a tool	Whether use is viable in the current case
Static policy rules	Known prohibitions and mandatory steps	Novel combinations, pressure regimes, missing evidence, and recovery loss
Model confidence thresholds	Uncertainty in the model output	Domain-level viability and downstream operational effects
Human-in-the-loop review	Final approval in theory	Workflow reliance, rubber-stamping, overload, and de facto authority
Observability/logging	What happened after action	Whether the action should have been admitted before execution
Risk scoring	Ranking or prioritizing cases	Non-substitutable domain thresholds and corridor preservation

The failure mode is usually not that one control is useless. The failure is that controls are stacked without a governing interpretation of viability. A system may have access control, logging, confidence scores, rules, and human review, yet still allow an agent to move risk from a visible domain into a hidden one.

6. The PV-PP Agent Governance Runtime

The PV-PP Agent Governance Runtime is a runtime decision layer. It evaluates proposed agent actions before execution and classifies them into governance outcomes. It can be implemented as a middleware service, an orchestration plugin, a policy-check API, a workflow gate, or a structured audit layer depending on the deployment environment.

Input to the Runtime	Governance interpretation
Proposed action or tool call	What the agent is trying to do and what authority it would exercise
Task context and user objective	What productive value the action is intended to create
Source evidence and source hierarchy	Whether the agent has enough authoritative evidence
Memory and state history	Whether prior obligations, failures, warnings, or commitments matter
Permissions and tool scope	Whether the available tool exceeds the viable authority corridor
Governing domains	Which domains cannot be sacrificed for local task success
Recovery conditions	Whether errors can be detected, reversed, escalated, or contained
Metric pressure	Whether local performance incentives are distorting admission

Runtime output	Meaning
Admit	The action is viable under current conditions.
Admit with constraints	The action may proceed only with scope, permission, logging, confirmation, or source constraints.
Escalate	The action is not viable for autonomous handling and must move to a human, specialist, or higher-control process.
Defer / request evidence	The agent lacks enough authoritative state to act.
Repair first	A missing source, stale state, broken prerequisite, or incomplete workflow must be corrected before action.
Reject	The action violates a hard governing domain or destroys a required recovery corridor.

7. Runtime Patterns

The runtime becomes practical when expressed as reusable runtime patterns. The following are the most important initial patterns.

Pattern	Function
Eligibility Gate	Determines whether the proposed action belongs in the admissible action set under current state, permissions, and governing-domain constraints.
Source-Authority Gate	Requires the agent to distinguish authoritative records from notes, summaries, retrieved snippets, historical workarounds, or model guesses.
Memory/State Check	Detects stale state, prior promises, prior failures, changed conditions, incomplete obligations, and misleading continuity.
De Facto Authority Check	Treats labels such as ready, resolved, approved, routine, or low risk as governing actions when downstream actors rely on them.

Adequacy Gate	Asks whether the action preserves the relevant task, evidence, correction, escalation, rollback, containment, and defensibility corridors.
Escalation Trigger	Routes ambiguous, high-stakes, conflict-of-source, out-of-scope, or near-threshold cases out of autonomous handling.
Guarded Re-Entry	Allows a previously failed, risky, or defecting actor/tool/source to re-enter only under constrained scope, monitoring, and evidence conditions.
Closure Verification	Prevents the agent from marking a workflow complete before asynchronous or downstream effects are confirmed.
False-Success Check	Detects whether dashboard gains could be hiding domain damage or future recovery costs.

8. Relationship Between the Agent Auditor and the Runtime

The PV-PP Agent Auditor and the PV-PP Agent Governance Runtime are related but distinct products.

Product	Role	Use case
PV-PP Agent Auditor	Structured review assistant that evaluates an agent design or deployment description.	Pre-deployment audit, risk review, sales discovery, architecture critique, internal governance review.
PV-PP Agent Governance Runtime	Runtime or near-runtime control layer that evaluates proposed actions and tool calls.	Live admission control, escalation, constrained execution, workflow gating, de facto authority control.

The Auditor is the easier public entry point. It asks structured questions and produces a governance report. The Runtime is the deeper product. It converts the same governance logic into runtime controls. The right adoption path is likely audit first, runtime later: use the Auditor to identify the risk surface, then implement the Runtime where the action surface justifies runtime control.

9. Where This Matters First

The runtime is most useful where agent outputs affect operational state, rights, money, legal position, privacy, trust, or recovery capacity. It is less useful for low-stakes drafting, ordinary summarization, and reversible internal assistance.

High-fit domain	Why it fits
Financial operations	Payment readiness, wire exceptions, fraud escalation, sanctions, client trust, irreversible loss.
Legal and e-discovery workflows	Privilege risk, production deadlines, provisional labels becoming practical release decisions.
Healthcare administration	Eligibility, documentation, patient communication, privacy, escalation, auditability.
Insurance and claims	Coverage classification, denial/approval signals, appeal rights, fraud review, regulatory exposure.

Enterprise IT and security	Access changes, incident triage, remediation authority, rollback, blast-radius control.
Customer operations	Refunds, cancellations, account changes, dispute paths, prior promises, trust damage.

The strongest first demo remains a wire-transfer exception review agent. It has clear stakes, clear workflow pressure, clear false-success metrics, clear source-authority hierarchy, and clear de facto authority risk. It is concrete enough to show the runtime's value without requiring a full enterprise deployment.

10. Benchmark Support and Limits

The PV-PP project now includes a substantial benchmark and simulation record. That record supports the architecture in three ways. First, it shows that the PV-PP framework can be made executable rather than remaining only a theoretical vocabulary. Second, it exposes recurring implementation patterns such as graph-bound policy construction, typed memory, cooperation propensity, guarded re-entry, adequacy gating, and false-success detection. Third, it distinguishes viability-preserving behavior from simple single-score optimization in constrained cases.

The benchmark record should not be overstated. Benchmarks are engineering evidence, design pressure, and demonstration material. They do not by themselves prove a universal theorem, certify deployed systems, or show that scalar aggregation is impossible. The proper claim is narrower and stronger: the benchmark program supports the practical viability of a corridor-preserving governance architecture and identifies mechanisms that should be carried into agent runtime design.

Benchmark-derived lesson	Agent-governance implication
Typed memory shapes future decisions	Agent state must include prior obligations, failures, promises, stale assumptions, and source history.
Cooperation propensity is relational and context-bound	Tool/subagent trust should not be a flat score; it should be scoped and history-aware.
Guarded re-entry prevents false rehabilitation	A failed actor/tool/source can regain limited use without regaining full authority.
Graph-bound policy generation controls candidate space	Tool/action candidates must be reachable, bounded, and explicitly represented.
Adequacy gates matter near pressure	Recovery, escalation, and rollback corridors must be checked before action, not after failure.
Scalar success can hide domain collapse	Dashboard improvement is not enough; hidden governing domains require explicit monitoring.

11. Adoption Path

The practical adoption path should avoid selling the full theory first. Organizations do not need to accept the whole PV-PP framework to benefit from the runtime. They need a way to identify when agent actions are viable rather than merely available.

1. Start with an audit. Use the PV-PP Agent Auditor to review one proposed or deployed agent workflow.

2. Map the action surface. Identify tools, permissions, labels, write actions, escalation points, closure states, and source dependencies.
3. Identify governing domains. Determine which domains cannot be traded away for speed or cost reduction.
4. Install narrow gates. Begin with source-authority checks, de facto authority checks, closure verification, and escalation triggers.
5. Add runtime state. Incorporate memory, prior failures, prior obligations, stale-data checks, and pressure indicators.
6. Instrument false success. Track downstream reversals, complaints, fraud, rework, audit exceptions, trust decay, and recovery cost, not just throughput.
7. Expand only after evidence. Move from advisory auditing to runtime controls where the workflow justifies it.

12. Competitive Positioning

The PV-PP runtime should not be positioned as another general AI platform. That category is crowded and vague. It should be positioned as a viability-control layer for agentic workflows.

Not the claim	Better claim
We make agents smarter.	We help decide when agent action is viable.
We replace orchestration.	We govern orchestration outputs before they create operational state.
We are a better rules engine.	We add corridor, memory, recovery, and domain-preservation logic around rules.
We prove agents are safe.	We expose viability gaps, false-success risks, and missing recovery corridors.
We solve all AI governance.	We target operational agent governance where tools, state, and authority matter.

The strongest phrase remains: available is not viable. That is the market wedge. Tool ecosystems are exposing availability faster than organizations are building viability governance. The PV-PP runtime fills that gap.

13. Limits and Honest Boundaries

The framework should not overclaim. The following boundaries should remain explicit:

- The runtime does not guarantee safety; it improves governance discipline around action admission.
- The benchmark program supports the architecture but does not close all formal proof questions.
- PV-PP terminology must be translated for external readers; operator-stack detail should be layered, not front-loaded.
- Runtime implementation requires real source hierarchies, permissions, logging, monitoring, and escalation channels. A conceptual gate without system integration is insufficient.
- Human review is not automatically adequate. If human review is overloaded, shallow, or dependent on agent labels, it may be part of the risk rather than the solution.
- The first commercial target should be narrow. Trying to sell a universal governance theory before a concrete agent workflow will dilute the message.

14. Conclusion

AI agents need more than access to tools. They need disciplined action admission. The fact that a tool exists, an API permits a call, a model is confident, or a workflow can be completed does not mean the action is viable. Viability depends on governing domains, source authority, memory, pressure, permissions, recovery, escalation, and downstream effects.

The PV-PP framework provides a language and architecture for that control problem. The Agent Auditor turns the logic into a structured review process. The Agent Governance Runtime turns the same logic into a runtime or near-runtime governance layer. Together, they create a practical path from theory to agent deployment: audit the risk surface, identify viability gaps, install corridor-preserving gates, and expand only where the workflow requires stronger control.

The immediate next step is not a broader claim. It is a concrete demonstration. A wire-transfer exception review agent is the best candidate because it makes the central distinction unavoidable: the relevant question is not whether the agent can mark the wire ready. The relevant question is whether "ready" is viable under the governing conditions.

Project Reference Set

This draft is based on the current PV-PP project repository and the recently created agent-governance support files, including:

- PV-PP Framework Project Index and Reader Guide v0.1
- PV-PP Framework Benchmark Extraction Insertion Note v0.1
- PV-PP Framework Benchmark-Derived Architecture Lessons and Implementation Patterns v0.1
- PV-PP Framework Runtime Implementation Pattern Catalog for Agent/Tool Governance v0.1
- PV-PP Agent Governance Runtime Product and Technical Roadmap v0.1
- PV-PP Framework Typed Memory as Governance-Shaping State v0.1
- PV-PP Framework Cooperation Propensity and Guarded Re-Entry v0.1
- PV-PP Agent Auditor System Instructions and Example Audit materials
- PV-PP Core Framework, Operators, Stack/Governance, Benchmarks, Proof, and Memory Model compendiums