

Human-in-the-Loop Control Planes for Cortex Agents: Policy-Driven Escalation, Approval, and Evidence Capture

Sashank Siwakoti¹

¹ Independent Researcher

Email: sashanksiwakoti1@gmail.com

Bhaskar Chaganti²

² Independent Researcher

Email: chagantibhaskar@gmail.com

Abstract: Purpose. Enterprise agents can accelerate anomaly triage and regulated reporting, yet ordinary access control, prompt instructions, and post-hoc traces do not define a complete authorization contract for consequential actions. This study develops PAVE-CP, a vendor-neutral control plane that determines when an agent must escalate, which evidence must accompany the request, who may approve, and how the resulting decision is enforced and recorded.

Design/methodology/approach. Following design-science research, requirements are derived from human-automation research, security principles, policy-as-code, provenance, AI-governance sources, recent agent-authorization work, and documented Cortex Agents interfaces. The artifact comprises a canonical action envelope, policy decision point, graded evidence model, approval broker, exact-action tokens, governed execution gate, postcondition verification, and a tamper-evident event ledger. Evaluation combines one million randomized policy-state checks, 20,000 end-to-end authorization checks, seven seeded mutations, and a reproducible discrete-event simulation of one million proposed actions.

Findings. The reference controller produced no structural mismatch in the exercised randomized and end-to-end state spaces and detected all seven seeded mutations. Under the stated synthetic assumptions, PAVE-CP routed 16.54% of proposals to review, preserved 81.78% bounded autonomy, and produced 0.36 modeled high-impact unsafe commits per 10,000 proposals, compared with 129.79 for role-based autonomy and 14.92 for blanket human review. Median latency was 2.34 seconds, while the approval-bound 95th percentile remained 237.10 seconds.

Originality, implications, and limits. The contribution is an externally enforceable commit protocol, not another prompt guardrail or observability dashboard. It binds policy, evidence, human authority, exact parameters, freshness, separation of duties, execution, and audit evidence into one verifiable control object. The Cortex mapping shows how rapid read-only investigation and drafting can coexist with explicit authorization for regulated publication and high-impact state change. Quantitative results are synthetic design-science evidence, not production telemetry or a compliance claim.

Keywords: AI agents; Human-in-the-loop; Runtime authorization; Control plane; Policy-as-code; Evidence capture; Approval workflow; Auditability; Snowflake Cortex Agents; Regulated reporting; Anomaly triage

Research highlights

- Defines a platform-independent authorization contract for allow, deny, and role-qualified approval at the point of consequence.
- Introduces evidential closure: reviewers receive a policy-mandated, machine-verifiable evidence bundle rather than an unstructured rationale.

- Binds approvals to canonical action, evidence, and policy digests with expiry, conditions, separation of duties, and replay protection.
- Maps the control plane to documented Cortex Agents tool-use, permission-decision, custom-tool, versioning, and observability surfaces.
- Provides executable schemas, policy-as-code, invariant and mutation tests, and a reproducible one-million-action synthetic evaluation.

1 Introduction

Large language model agents move generative AI from recommendation toward execution. They can interpret an underspecified goal, select tools, retrieve and transform evidence, and create durable effects in databases, files, ticketing systems, communication channels, and reporting platforms. This capability is especially attractive in data-intensive enterprises, where anomaly investigation and report preparation consist of many small steps that are individually automatable but collectively expensive. It also changes the unit of control. A generated answer can be reviewed after the fact; a tool invocation may change a governed state before anyone realizes that an authorization decision was required.

Three familiar governance patterns are necessary but insufficient when used alone. Role-based or attribute-based access control determines which resources a principal may technically reach, but not whether a specific invocation is appropriate for the current purpose, workflow stage, data class, materiality, uncertainty, or expected blast radius [15–20,48]. Prompt instructions can influence a probabilistic planner but cannot guarantee complete mediation. Conversely, sending every non-trivial action to a person produces queueing delay, approval fatigue, and perfunctory confirmation [10–14,32,39,40]. The practical objective is bounded autonomy: rapid execution inside a demonstrably safe envelope, with explicit human authority at risk-bearing boundaries.

Regulated workflows expose the missing contract most clearly. Meaningful oversight requires more than a confirmation dialog. A reviewer must be able to determine what exact action will occur, why it is proposed, which records and rules support it, what will change, whether the effect is reversible, and whether the operating context has drifted since the evidence was assembled. The system must then prove that the authorized transaction—not a

modified successor—was the one committed, and it must preserve a decision record suitable for later control testing.

Agent-security research has advanced substantially through tool-use benchmarks, red-team environments, prompt-injection studies, delegation models, runtime interception, and human-centered permission analysis [26–32,42–47]. Yet these strands do not by themselves specify a regulated-workflow commit protocol that composes policy evaluation, evidence sufficiency, role-qualified approval, separation of duties, exact-action binding, freshness, replay prevention, postcondition verification, provenance, and durable decision logging. The problem is particularly acute on managed agent platforms whose orchestration loop cannot simply be rewritten.

This article addresses that systems-design gap with PAVE-CP—the Policy, Approval, Verification, and Evidence Control Plane. PAVE-CP sits between an agent proposal and a consequential commit. It canonicalizes the proposed action, evaluates a versioned policy bundle, assembles and validates a tier-appropriate minimum evidence bundle, obtains any required approval tokens, revalidates all predicates immediately before execution, verifies postconditions, and writes a correlated event chain. Low-risk read, analysis, and drafting operations remain fast; regulated publication and high-impact state change cannot commit without the evidence and authority specified by policy.

Figure 1 summarizes the governing distinction. Agent reasoning remains probabilistic and adaptive, but the transition from proposal to authoritative effect is mediated by a deterministic authorization plane. This separation preserves speed before consequence and makes human authority meaningful only where policy defines it as part of the commit predicate.

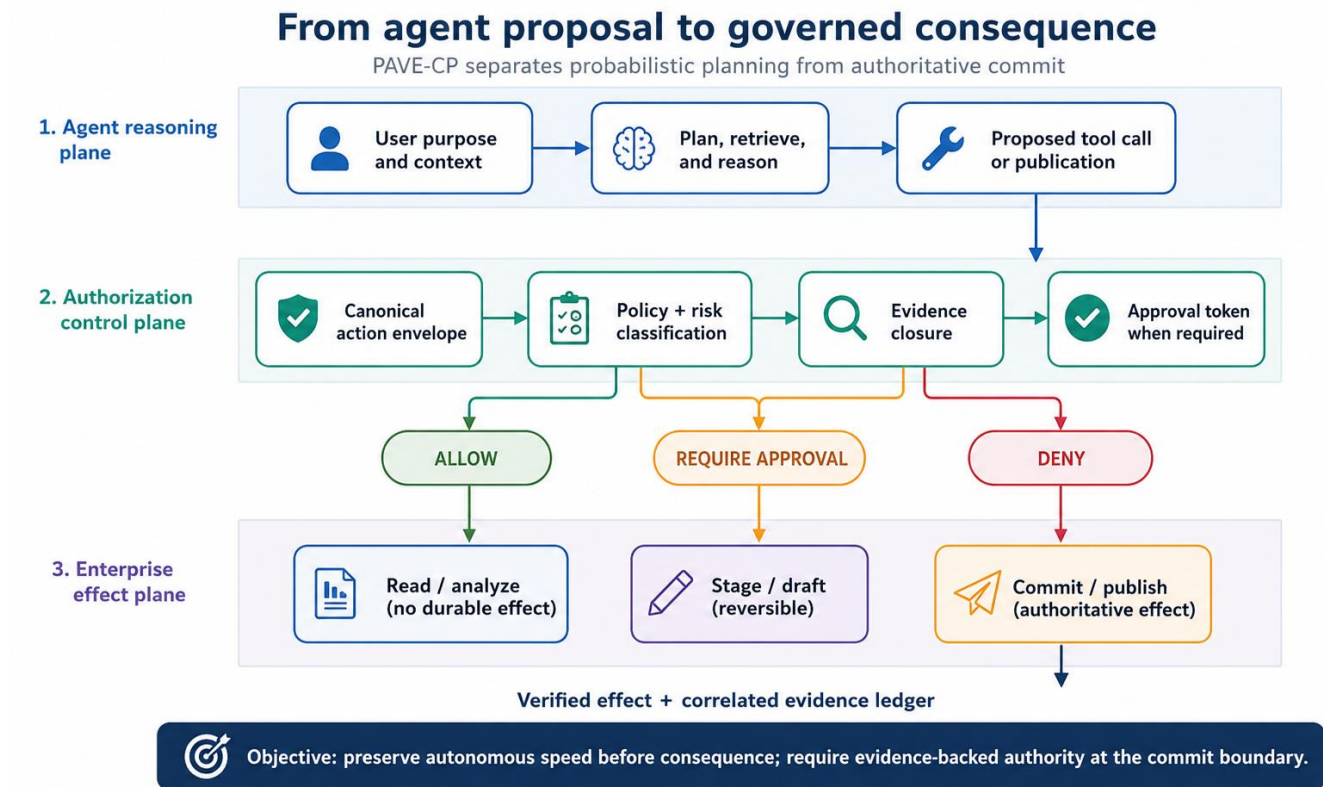


Figure 1. From agent proposal to governed consequence: the PAVE-CP control objective.

Source: authors' design.

The artifact is developed against Snowflake Cortex Agents but remains vendor-neutral. Cortex Agents provides a useful design target because it combines managed plan-tool-reflect orchestration, structured and unstructured data tools, custom procedures and functions, streamed tool-use events, permission decisions, versioned agent configurations, and AI observability [33–38]. This environment illustrates both the opportunity and the constraint: organizations can gain agentic speed inside a governed data platform, but they still need an organizational authorization layer that translates enterprise policy into evidence-backed, action-specific decisions.

The study asks three research questions. RQ1: How can approval obligations be expressed and enforced independently of prompts and individual tool implementations? RQ2: Which evidence, authority, and binding properties are necessary for auditable human oversight of consequential agent actions? RQ3: Under a controlled synthetic workload, can risk-adaptive escalation reduce modeled unsafe commits and review demand relative to role-based autonomy and blanket human review while preserving a low-latency autonomous path?

The article contributes: (1) a control-plane architecture and governed action lifecycle; (2) a formal authorization contract linking policy, evidence, approvals, execution, and audit; (3) a graded escalation and evidence model for anomaly triage and regulated reporting; (4) a concrete Cortex Agents integration pattern that relies only on documented platform surfaces; (5) six design principles with falsifiable invariants; and (6) an executable reference artifact with schemas, policy-as-code, mutation tests, queueing simulation, and sensitivity analysis. The claim is deliberately bounded: PAVE-CP is a testable design theory and reference implementation, not a claim that one technical component can establish regulatory conformity.

2. Background and related work

2.1 Agentic tool use and the authorization gap

Contemporary agent architectures interleave planning, tool selection, observation, and response. ReAct made this pattern explicit by combining reasoning and action trajectories [26], while AgentBench demonstrated the breadth of environments in which language models can be evaluated as agents [27]. The

pattern is powerful because the model can revise a plan after intermediate results. The control implication is that the final action sequence and all effect-bearing parameters are rarely known at request admission. Authorization must therefore be evaluated repeatedly at action boundaries, not only at login or at the initial user prompt.

Tool-capable agents face accidental and adversarial failure modes. LM-emulated sandboxes, AgentDojo, and Agent Security Bench show that agents can be redirected through untrusted data, prompt injection, memory manipulation, or unsafe tool composition [28,29,42]. The OWASP agentic-risk taxonomy similarly foregrounds goal hijacking, tool misuse, privilege abuse, memory poisoning, and cascading failures [47]. These findings support a core premise of PAVE-CP: a model-generated tool call is untrusted input to the execution layer even when it originates from an authenticated agent.

Identity and access management remain indispensable, but they answer a different question. A delegated agent may legitimately hold read and write privileges while a particular write remains outside the user's purpose, crosses an egress boundary, publishes a draft as final, or affects far more objects than anticipated. Complete mediation and attribute-based authorization therefore need a semantic action layer: subject, purpose, parameters, targets, data class, reversibility, expected effect, and policy version must be evaluated before commitment [15,16,18,48].

2.2 Human oversight without approval theatre

Human-in-the-loop is often treated as a binary property—either a person is present or the system is autonomous. Human-factors research instead emphasizes levels of automation, allocation of function, and the conditions under which people appropriately rely on or disuse automated advice [10–14,39,40]. The irony of automation is that reviewers may be left with the least frequent, most ambiguous, and highest-consequence interventions, even as their familiarity with the underlying process declines [11]. A confirmation step can therefore be nominally human-controlled while remaining practically ineffective.

Automation bias, poor calibration, and approval fatigue are not solved by adding more dialogs [12,32,39,43,45]. Reviewers are more likely to miss errors when an interface foregrounds a confident recommendation while hiding contradictory evidence,

and they may reject safe actions when uncertainty is poorly expressed. Commercial permission mechanisms also vary widely in scope, timing, explanation, and user control [45]. A defensible approval design must therefore reduce unnecessary review, make consequences and alternatives salient, and provide enough structured evidence for the reviewer to detect a material defect.

PAVE-CP operationalizes these insights in four ways. Escalation is selective rather than universal; the reviewer receives a compact, immutable decision packet rather than an unstructured transcript; decisions include reject, approve once, and approve subject to enforceable conditions; and every approval expires and binds to the exact proposed transaction. These mechanisms make review consequential, inspectable, and cognitively scoped. They also create measurable operational outcomes—review demand, queue latency, rejection reasons, evidence deficiencies, overrides, and post-approval exceptions—rather than reducing oversight to click counts.

2.3 Policy-as-code, provenance, and tamper evidence

A control plane separates policy decision from business execution. XACML formalized policy decision and enforcement points [18], while Open Policy Agent illustrates a contemporary policy-as-code pattern in which applications submit structured input to a declarative decision engine [19]. Decision logs can retain policy inputs, bundle metadata, and outcomes [20]. PAVE-CP adopts this separation but enriches the policy output with evidence obligations, approval cardinality, required roles, validity periods, reason codes, conditional constraints, and postcondition obligations.

Evidence must remain attributable as it moves among the agent, policy engine, reviewer interface, and executor. W3C PROV provides a vocabulary for entities, activities, and responsible agents [22]; W3C Trace Context and OpenTelemetry provide correlation identifiers for distributed execution [23,24]. PAVE-CP uses both forms of linkage. Source identifiers, query IDs, and lineage references explain where an assertion came from, while run, thread, tool-use, trace, and span identifiers explain where it occurred in the technical workflow.

Authorization also depends on stable representation. Canonicalization before hashing reduces ambiguity

between what a reviewer saw and what a gateway later executed; the JSON Canonicalization Scheme is one applicable mechanism [21]. Hash chaining and signatures do not make a ledger inherently immutable, but they can make unauthorized alteration detectable under explicit trust assumptions [25,41]. PAVE-CP applies those principles to control events and binds every approval to digests of the action, validated evidence, and active policy bundle.

2.4 Governance requirements and the research gap

Governance frameworks increasingly call for explicit accountability, traceability, lifecycle control, and suitable human oversight. NIST AI RMF maps risks to context and organizational responsibility [4], while the NIST Generative AI Profile adds actions concerning defined roles, oversight processes, and evidence of risk treatment [5]. The EU AI Act includes record-keeping and human-oversight duties for high-risk systems [6]. ISO/IEC 42001, ISO/IEC 23894, and ISO/IEC 5338 address AI management systems, risk management, and lifecycle processes [7–9]. These sources do not prescribe an agent gateway; together, however, they support the need for defined authority, reconstructable decisions, controlled change, and evidence-backed intervention.

The implementation gap is therefore not a lack of individual mechanisms. Enterprises already operate access-control systems, workflow engines, policy services, audit stores, observability platforms, data-lineage catalogs, electronic-signature processes, and agent runtimes. What is missing is a standardized transaction that composes those mechanisms at the precise moment

an agent crosses from analysis to consequence. Without such a transaction, approval can be detached from final parameters, evidence can be stale or incomplete, and a trace can show that a tool ran without showing why the action was authorized.

Recent work makes the boundary increasingly visible. AEGIS interposes a pre-execution firewall and tamper-evident audit layer on tool calls [31]; AARM specifies runtime interception, contextual policy evaluation, and action receipts [44]; authenticated-delegation work scopes the authority that people grant to agents [46]; Janus and landscape analyses examine user involvement and permission design [43,45]; and a systematic review frames agent security as an agent–human interaction problem [32]. PAVE-CP is complementary but more specific to regulated organizational workflows: it standardizes evidential closure, role and cardinality requirements, separation of duties, exact-action authorization, postcondition verification, and correlation with a managed platform.

2.5 Closest approaches and differentiated contribution

Table 1 positions PAVE-CP against four adjacent control families. The comparison is not intended to diminish those approaches: identity systems, runtime firewalls, delegation protocols, and permission interfaces are all useful components of a defense-in-depth architecture. The differentiated contribution is the unit of control. PAVE-CP treats one canonical transaction, its minimum evidence, the active policy, the required organizational authority, and the verified effect as a single auditable object.

Table 1. Relationship between PAVE-CP and adjacent agent-control approaches.

Approach	Primary control boundary	Evidence requirement	Human authority semantics	Binding / verification	Remaining gap addressed by PAVE-CP
RBAC / ABAC / IAM	Principal–resource access	Attributes and credentials; usually no action-specific evidence packet	Role or delegated scope	Access-token and resource checks; access logs	Does not determine semantic appropriateness, materiality, workflow stage, or exact transaction evidence [15–18,46,48].

Prompt and agent guardrails	Model instruction or generated content	Rationale or policy text; not independently validated	Usually advisory or absent	No reliable commit binding; model can be bypassed by tool paths	Cannot provide complete mediation or prove that an approved transaction was executed.
Runtime tool firewalls / specifications	Proposed tool invocation before execution	Risk features and session context; human escalation may be optional	Block, allow, or request approval	Pre-execution interception; tamper-evident logs or receipts in AEGIS/AARM [31,44]	Evidence sufficiency, multi-role approval, exact evidence/policy binding, and postconditions are not standardized.
Delegation and permission systems	Delegated capability, tool, or user-facing request	Scope and user-facing context	User grants, narrows, or denies authority	Tokens or permission decisions vary by system [43,45,46]	Regulated organizational approval, separation of duties, and control evidence are not the central unit.
PAVE-CP	Canonical action at the commit boundary	Tier-specific minimum evidence bundle with freshness and provenance	Role-qualified k-of-n approval, conflicts, conditions, and expiry	Action/evidence /policy digests, single-use nonce, precondition and postcondition checks	Requires domain calibration, secure implementation, and field validation; it does not itself establish compliance.

Note. The table summarizes primary design emphasis rather than claiming that every implementation in a category has identical features. AEGIS, AARM, delegation, and permission research are cited in the relevant cells.

This positioning yields two boundary conditions. First, PAVE-CP assumes that the executor can be mediated or replaced by a governed proxy; where this is impossible, the design can observe but cannot guarantee prevention. Second, the artifact standardizes technical predicates, not the substantive legal or domain judgment that defines risk tiers, competent reviewers, or sufficient evidence. Those remain governed organizational inputs.

3. Research method

3.1 Design-science strategy and unit of analysis

Design-science research is appropriate because the primary output is an evaluated artifact intended to solve a coupled organizational and technical problem [1–3]. The

problem occurs at the boundary between agent autonomy and regulated effect: prevailing controls do not consistently express when an agent must escalate, what a reviewer must see, or what must be retained to reconstruct the decision. The artifact includes constructs, design principles, architecture, schemas, policy semantics, an executable authorization core, and a managed-platform integration pattern.

The unit of analysis is one proposed consequential action at one commit boundary. Conversations, plans, and model responses provide context, but they are not the authorization object. This choice makes the design testable: each proposal either satisfies a defined commit predicate or ends in a reason-coded denial, expiry,

cancellation, exception, or verified execution.

3.2 Relevance and rigor cycles

The relevance cycle uses two reference workflows. Anomaly triage contains a high volume of read, query, compare, and draft steps, punctuated by occasional state changes such as quarantine, notification, or case creation. Regulated reporting contains evidence collection, reconciliation, calculation, narrative generation, review, attestation, and authoritative publication. The workflows share tools but differ in materiality and the consequences of premature commitment, making them suitable for testing risk-adaptive control.

The rigor cycle synthesizes human-automation allocation and appropriate reliance [10–14,39,40]; complete mediation, least privilege, zero trust, and attribute-based authorization [15–17,48]; policy-as-code and decision logging [18–20]; provenance, trace correlation, canonicalization, and tamper evidence [21–25,41]; agent tool-use risk and runtime authorization [26–32,42–47]; and AI-governance sources [4–9]. Governance materials are used to derive control concerns, not to assert legal conformity.

3.3 Artifact construction and evaluation logic

Construction proceeded from requirements to design principles, a formal decision contract, component architecture, state machine, risk and evidence tiers, Cortex integration mapping, and an executable reference implementation. The implementation uses deterministic canonical JSON, SHA-256 digests, HMAC-signed demonstration tokens, role and cardinality checks, evidence freshness, atomic nonce consumption in the test model, and a hash-linked event ledger. HMAC and in-memory examples are teaching mechanisms; production deployments require managed key custody, durable distributed state, and independently protected retention.

Evaluation triangulates three forms of evidence. First, structural property tests generate 1,000,000 randomized policy states and check escalation invariants. Second, 20,000 end-to-end authorizations exercise action, policy,

evidence, approval, and commit bindings; seven mutation tests deliberately remove or corrupt a required control. Third, a discrete-event simulation compares RBAC-only autonomy, blanket human review, and PAVE-CP across 20 replications of 50,000 proposals. This sequence tests internal coherence before exploring operational trade-offs.

The quantitative criteria are approval request rate, denial rate, bounded-autonomy rate, modeled unsafe commits per 10,000 proposals, modeled high-impact unsafe commits per 10,000, median/mean/95th-percentile latency, executed-action evidence completeness, false rejection of safe proposals, and policy-decision overhead. Brackets in the results are the 2.5th and 97.5th percentiles across 20 replications. They are empirical replication intervals, not inferential confidence intervals for real deployments.

3.4 Validity posture, reproducibility, and ethics

No human participants, personal data, production records, or live regulated decisions were used. The evaluation supports internal validity of the implemented bindings and comparative behavior inside the stated model; it does not establish external validity for an industry, organization, regulation, reviewer population, or Cortex deployment. All parameters, schemas, tests, random seeds, result files, and figure-generation code are included in the reproducibility supplement so that reviewers can inspect assumptions and regenerate outcomes.

4. Design requirements

Table 2 translates broad governance and security concerns into testable design requirements. Table 3 then converts those requirements into design principles and falsifiable propositions. A deployment may strengthen thresholds or add domain-specific evidence, but weakening complete mediation, evidential closure, or exact binding should be treated as a documented change to the safety case rather than an implementation convenience.

Table 2. Traceability from governance concerns to PAVE-CP design requirements.

Requirement	Operational specification	Primary grounding
R1 Complete mediation	Every consequential action and publication attempt passes through a policy enforcement point immediately before commit.	Complete mediation; AI risk controls; tool-call risk [4,5,15,28,29].
R2 Risk-adaptive escalation	The decision engine returns ALLOW, REQUIRE_APPROVAL, or DENY using action impact and contextual triggers.	Human-automation allocation; proportionate oversight [6,10,14].
R3 Fail-closed obligations	Missing policy, evidence, identity, or authorization state prevents commitment; degraded modes are explicit.	Least privilege; zero trust; security controls [16,17].
R4 Evidence sufficiency	Each tier has a machine-checkable minimum evidence bundle with freshness and provenance.	Traceability, logging, risk documentation [4–9,22].
R5 Exact-action binding	Approval is cryptographically bound to canonical action, evidence, and policy digests.	Integrity, accountability, prevention of approval drift [21,25].
R6 Qualified independence	Cardinality, approver roles, requester-approver conflict, and separation of duties are enforced.	Defined human authority and effective oversight [5,6,10].
R7 Freshness and replay safety	Decisions and evidence expire; nonces and idempotency keys prevent repeated use; commit rechecks drift.	Transaction integrity and zero-trust verification [16,17,21].
R8 Reconstructable ledger	Policy inputs, outcomes, evidence references, approvals, execution, verification, and exceptions share correlation IDs.	Decision logging, provenance, observability [20,22–24,35,36].
R9 Verified effect	Staged execution, parameter constraints, affected-object checks, and postconditions precede irreversible finalization where feasible.	Risk treatment and runtime assurance [4,8,28,31].
R10 Human-centered review	The interface presents differences, sources, effect, uncertainty, rollback, conditions, and reason codes without relying on private chain-of-thought.	Appropriate reliance; usable human-AI interaction [12–14,30,39].
R11 Platform interoperability	The control object is independent of a particular model and maps to available tool, permission, custom-action, and trace interfaces.	Lifecycle management and policy separation [9,18,19,33–38].
R12 Measurable governance	The system exposes approval burden, queue delay, evidence failure, overrides, policy drift, and control exceptions as operational metrics.	Monitoring, evaluation, continual improvement [4,5,7,36].

Note. Requirements are derived from the cited governance, human-factors, access-control, and agent-security literature; they are technical design obligations rather than legal conclusions.

Table 3. PAVE-CP design principles and falsifiable propositions.

Design principle	Operational mechanism	Falsifiable proposition / test
DP1 Complete mediation	All consequential tool calls and publications pass through a trusted PEP immediately before effect.	P1: no governed executor accepts an action without a correlated authorization or reason-coded denial.
DP2 Risk-adaptive allocation	Effective tier is the maximum of base impact and deterministic contextual triggers.	P2: low-risk actions remain autonomous while mandatory high-impact actions cannot bypass review.
DP3 Evidential closure	The PDP emits obligations; review begins only after evidence is complete, attributable, valid, and fresh.	P3: no action reaches PENDING_APPROVAL or AUTHORIZED with an unsatisfied mandatory obligation.
DP4 Exact-action binding	Canonical action, evidence, and policy digests are embedded in expiring, single-use tokens.	P4: action drift, evidence drift, policy drift, stale tokens, and replay are rejected at commit.
DP5 Qualified independent authority	Role coverage, cardinality, conflicts, and separation of duties are machine checked.	P5: approval sets that lack distinct qualified reviewers never satisfy the commit predicate.
DP6 Closed-loop assurance	Execution is idempotent or explicitly bounded; postconditions and a correlated event chain close the action.	P6: every terminal action has reconstructable decision, execution, verification, and exception evidence.

Note. A proposition is falsified when a conforming implementation permits the prohibited state or cannot produce the specified control evidence.

The requirements deliberately distinguish evidence from explanation. A natural-language rationale can help orient a reviewer, but it is neither necessary nor sufficient for authorization. The mandatory evidence is structured and externally inspectable: identities, exact parameters, source and lineage references, diffs, validation outcomes, expected effects, rollback information, timestamps, and content digests. This distinction avoids treating a model's private reasoning trace as an audit record, particularly because generated chain-of-thought may be incomplete or unfaithful [30]. The principles in Table 3 make this distinction operational.

5. PAVE-CP artifact design

5.1 Architectural overview

PAVE-CP consists of five cooperating planes, summarized in Figure 2. The decision plane canonicalizes the action and evaluates policy. The evidence plane collects and validates tier-specific facts. The approval plane routes an immutable decision packet to qualified reviewers and returns signed tokens. The execution plane enforces authorization through a governed adapter or proxy. The assurance plane verifies postconditions,

correlates platform traces, and records the complete lifecycle in an integrity-protected ledger. The agent proposes work; it is never the authority that permits its own consequential action.

The policy enforcement point (PEP) is the trusted choke point. It receives a proposal before side effects occur, constructs the action envelope, and prevents direct access to governed executors. The policy decision point (PDP) returns a structured decision object rather than a bare Boolean. The object contains effect, effective risk tier, reason codes, minimum evidence obligations, approval cardinality and roles, validity period, enforceable conditions, postcondition requirements, and policy version/digest. This makes the decision executable by downstream components and reconstructable after the fact.

The evidence builder retrieves only what is required for the current decision. It may call lineage services, dry-run endpoints, schema validators, reconciliation queries, diff generators, data-classification services, or business-control catalogs. The validator checks field presence, source locators, hashes, collection time, and alignment

with the current action digest. It does not ask the agent to disclose private chain-of-thought. A concise model-generated rationale may be attached as non-authoritative context, but the decision packet must stand on independently inspectable evidence.

The approval broker resolves eligible reviewers using organizational roles, workload, conflict-of-interest rules, delegation status, jurisdiction, and availability. Each reviewer sees the same immutable packet. A completed review becomes a scoped token only after identity, role, decision, and packet bindings are validated. The token is not a durable grant to a tool or action class; it is a single-use authorization for one action digest, one evidence

digest, one policy digest, one tool-use identifier, and one validity interval.

The execution adapter accepts only authorized commit objects. It re-canonicalizes the action, checks all digests and signatures, verifies role coverage and separation of duties, consumes nonces atomically, and compares current preconditions with the approved snapshot. Where the target supports staging or transactions, the adapter enforces constraints and evaluates postconditions before final commit. Where it does not, the adapter minimizes privilege, records the weaker atomicity guarantee, and captures compensation and rollback status.

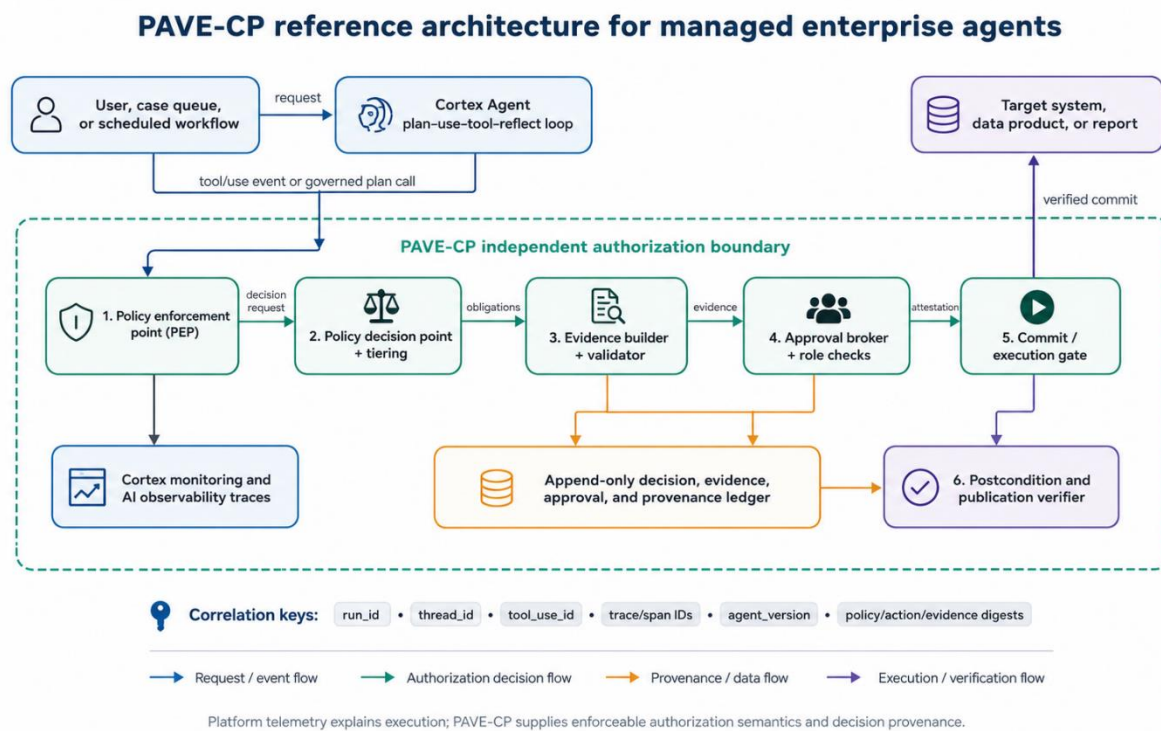


Figure 2. PAVE-CP reference architecture and governed action lifecycle.

Source: authors' design. Dashed boundary denotes the independent authorization trust boundary.

5.2 Action envelope and decision function

An action envelope is the smallest complete object on which policy can operate. It is more specific than a user prompt and more stable than a rendered review screen. Required fields include run, thread, tool-use, agent and agent version, requester, purpose, action type, tool, parameters, targets, data classifications, external-egress indicator, privilege-change indicator, reversibility, estimated impact, uncertainty, proposed time, and idempotency key. Domain extensions may add legal entity, jurisdiction, report period, control owner, transaction value, patient context, or model-inventory

identifiers.

Let a denote the action envelope, c the external context, and P the active policy bundle. Canonicalization C14N maps a structured object to one deterministic byte representation. The action and policy digests are:

$$h_a = H(C14N(a)); \quad h_p = H(P) \quad (1)$$

The policy decision function returns more than permission. It returns an effect d , normalized tier t , evidence obligations O , approval cardinality k , required role set R , validity interval τ , reason codes Z , and postcondition set V :

$\text{delta}(P, a, c) \rightarrow (d, t, O, k, R, \text{tau}, Z, V)$ (2)

Context can include data classification, workflow stage, user role, target environment, business calendar, outstanding incidents, prior denials, anomaly score, uncertainty, or a policy-specific risk signal. Policy authors must distinguish deterministic hard triggers from

probabilistic indicators. Restricted-data egress, privilege change, regulated publication, prohibited targets, and irreversibility can be hard triggers. A learned risk score may raise a tier or require more evidence, but it must not silently lower a mandatory control.

5.3 Risk and evidence tiers

Table 4. Default risk tiers, evidence grades, and escalation effects.

Tier	Typical effect	Minimum evidence grade	Default decision
T0 Observe	Read-only metadata/status; no sensitive export; no authoritative output	E0 identity, action/policy digests, trace context	Allow unless a hard trigger applies
T1 Analyze	Read/query/compare; ephemeral calculation; internal summary	E1 + source identifiers, lineage, expected effect	Allow; sample or escalate on contextual trigger
T2 Prepare	Draft report, create non-authoritative case, reversible low-impact update	E2 + parameter diff, validation checks, rollback plan	Allow or one approval when regulated/restricted/suspicious
T3 Commit	Material state change, controlled notification, authoritative internal release	E3 + dry run, affected-object count, independent review	One qualified approval
T4 Publish	External filing, irreversible or high-blast-radius action, cross-boundary disclosure	E4 + dual attestation, separation of duties, final reconciliation	Two distinct role-qualified approvals
T5 Prohibited	Policy-forbidden action, unsupported jurisdiction, uncontrolled secret/credential use	Record denial evidence and reason codes	Deny; no override token in normal path

The tiers in Table 4 describe potential consequence, not task complexity. A sophisticated analytical query can remain T1 when it is read-only and contained; a simple send or publish action can be T4 because it crosses an authoritative boundary. The effective tier is the maximum of the base tier and all mandatory contextual escalations. An irreversible action, for example, cannot remain below T3 merely because its estimated monetary value is low.

Policy should be compositional. A base rule sets tier

from action type and target. Data rules add obligations for restricted classes. Workflow rules prevent publication before reconciliation. Identity rules determine reviewer eligibility and conflicts. Runtime rules handle freshness, drift, and accumulated sequence risk. Figure 3 shows how these components resolve to allow, approval, or denial only after evidential closure. A separately governed break-glass route may exist, but a generic allow-on-error fallback violates the fail-closed requirement.

Risk-adaptive escalation and evidence-closure decision flow

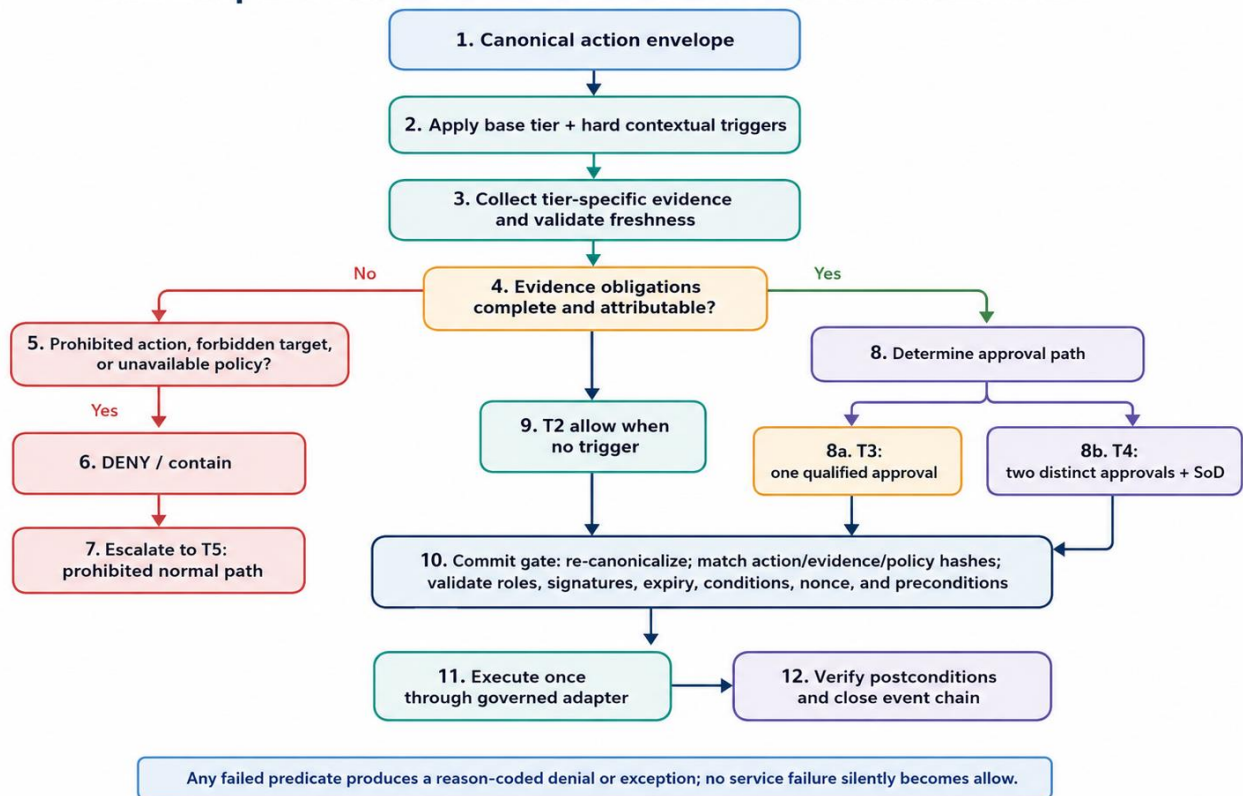


Figure 3. Risk-adaptive escalation and evidence-closure decision flow.

Source: authors' design.

5.4 Minimum evidence bundle

Table 5. Standard minimum evidence bundles (MEB).

Evidence group	Illustrative fields	Control purpose
Identity and purpose	requester, agent, version, workflow, purpose, run/thread/tool-use IDs	Establishes accountable actors and decision scope.
Canonical action	action type, tool, exact inputs, targets, data classes, idempotency key, h_a	Prevents an approval screen from abstracting away material parameters.
Policy state	decision ID, effect, tier, reasons, obligations, policy version and h_p	Makes the applied rule set reconstructable.
Source and lineage	source IDs, query IDs, record/time ranges, citations/locators, content hashes	Supports traceability from assertion to governed data.
Effect and reversibility	before/after diff, expected effect, blast radius, rollback/compensation	Shows what will change and how failure can be contained.
Validation	schema checks, reconciliations, dry run, affected rows/objects, constraint results	Supplies machine-checkable preconditions and review cues.
Uncertainty	confidence, anomaly/risk signals, unresolved exceptions, assumptions	Supports appropriate reliance and policy escalation.
Trace and	trace/span IDs, collected-at/expires-at, dependency	Correlates distributed events and

freshness	versions	prevents stale evidence.
Integrity	evidence digest h_e , field-level hashes, evidence-builder identity/signature	Binds the reviewed packet to commit authorization.

Evidence is graded rather than uniformly maximal. Excessive evidence can be as damaging as insufficient evidence because it obscures salient defects and enlarges queues. The PDP therefore returns obligations O , and the validator evaluates $\text{Complete}(e, O)$. A T0 status query does not need a rollback plan; a T4 filing does. Policies may add content-specific obligations such as reconciliation to an authoritative ledger, a signed source extract, a dual-control checklist, recipient validation, or an explicit list of unresolved exceptions. Table 5 defines the standard minimum evidence bundle from which these tier-specific packets are composed.

The evidence digest is computed after validation, not before. Let e be the normalized evidence object and $h_e = H(C14N(e))$. Any later change to a source locator, parameter diff, validation result, or expiry produces a new digest and invalidates existing approvals. Evidence can reference large artifacts by content hash and durable URI rather than embedding them in the token. Access to those artifacts must be preserved for the audit retention period and protected from unauthorized alteration.

$\text{Complete}(e, O) = 1$ iff every mandatory obligation in O is present, valid, attributable, and fresh (3)

5.5 Approval token and exact-action authorization

A reviewer decision becomes an approval token only after identity, role, decision, and packet bindings are

validated. The token contains a token ID, decision ID, tool-use ID, h_a , h_e , h_p , approver identity and roles, decision choice, enforceable conditions, issue time, expiry, and a unique nonce. It is signed by an organizational approval service or through a trusted reviewer-signing workflow. The demonstration uses HMAC for simplicity; production deployments should use managed keys and, where independent attestation is required, asymmetric signatures or equivalent enterprise signing mechanisms.

For approver i , a simplified token is:

$T_i = \text{Sign}_{sk_i}(\text{run_id}, \text{tool_use_id}, h_a, h_e, h_p, \text{choice}, \text{conditions}, \text{exp}, \text{nonce})$ (4)

The commit predicate evaluates the current action, current evidence, active policy, approval-token set, time, and execution preconditions. A proposal commits only when policy permits it, evidence obligations are complete, the required number and roles of reviewers are present, prohibited conflicts are absent, each token is valid and correctly scoped, all hashes still match, the decision and evidence are fresh, nonces are unused, and preconditions have not drifted. Figure 4 depicts this exact-action binding.

$\text{Commit}(a)$ iff $d \neq \text{DENY}$ and $\text{Complete}(e, O)$ and $\text{ValidApprovals}(A, k, R)$ and Fresh and Bound and NotReplay and PreconditionsHold (5)

Exact-action authorization binding and single-use commit

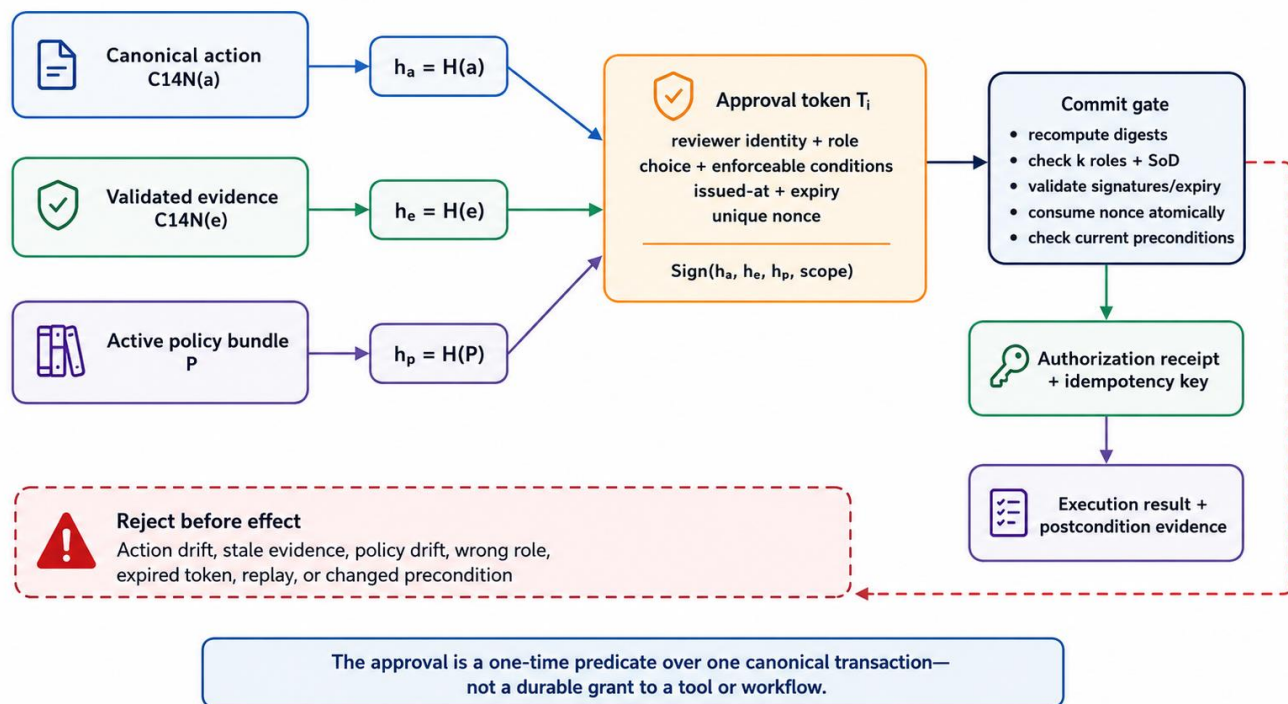


Figure 4. Exact-action binding from canonical proposal and evidence to a single-use approval token and verified commit.

Source: authors' design.

Proposition 1 (action non-substitutability). Assume C14N is deterministic, H is collision resistant for the deployment's threat model, signatures cannot be forged, and the PEP is trusted. A valid approval for action a cannot authorize a materially different action a-prime whose canonical representation differs, because h_a will differ and the commit gate will reject the token binding. The proposition does not protect against a malicious semantic mapping that hides a material parameter before canonicalization; action schemas and policy adapters must therefore expose all effect-bearing fields.

Proposition 2 (single-use authorization). Assume nonce consumption is atomic and durably shared by all execution-gate replicas. After a token nonce is consumed for an authorized commit, a replay using the same token

is rejected. High availability must not weaken this assumption: local in-memory nonce stores are insufficient for multi-replica production deployment.

Proposition 3 (tamper evidence). Assume each ledger event includes the prior event digest and is signed or committed to independently protected storage. Alteration or deletion inside the chain is detectable at verification, although prevention and non-repudiation depend on key custody, retention architecture, access controls, and external anchoring. PAVE-CP therefore describes a tamper-evident ledger, not an inherently immutable one.

5.6 State machine, invariants, and exception handling

Table 6. PAVE-CP governed action state machine.

State	Entry condition	Permitted transition
PROPOSED	Action envelope accepted and canonicalized	EVALUATED or FAILED
EVALUATED	Policy decision and obligations issued	ALLOWED, PENDING_APPROVAL, or DENIED

PENDING_APPROVAL	Evidence complete; packet awaiting required reviewers	APPROVED, REJECTED, EXPIRED, or CANCELLED
APPROVED	Required valid tokens exist for current hashes	AUTHORIZED or REVALIDATION_REQUIRED
AUTHORIZED	Commit gate atomically consumes authorization	EXECUTED or EXECUTION_FAILED
EXECUTED	Action staged or applied; actual effect captured	VERIFIED, COMPENSATING, or EXCEPTION
VERIFIED	Postconditions and publication checks pass	CLOSED
DENIED/REJECTED/EXPIRED	No commit; reason and evidence retained	CLOSED or NEW_PROPOSAL

Table 7. Commit-time invariants and required failure evidence.

Invariant	Commit-time predicate	Fail-closed response	Audit evidence
I1 Complete mediation	Executor request carries a valid authorization receipt for the current action.	Reject; raise bypass event; disable direct path.	Proposal, policy decision, authorization receipt, executor identity.
I2 Tier enforcement	Decision effect, tier, and hard triggers are consistent with active policy.	Deny or re-evaluate under current bundle.	Tier derivation, reason codes, policy version/digest.
I3 Evidential closure	$\text{Complete}(e,O)=1$ and all evidence is attributable and fresh.	Return reason-coded remediation; no approval request.	Obligation checklist, source IDs, hashes, timestamps.
I4 Exact binding	Current h_a , h_e , and h_p equal every approval-token binding.	Reject as drift or substitution.	Canonical objects and digest comparison.
I5 Freshness and replay safety	Decision, evidence, and tokens are unexpired; nonce is unused.	Reject; atomically mark/retain replay evidence.	Trusted time, expiry, nonce-consumption record.
I6 Independent authority	Distinct reviewer identities satisfy k , required roles, and conflict rules.	Reject incomplete or conflicted approval set.	Identity, role resolution, delegation, conflict check.
I7 Lifecycle completeness	Every terminal state has correlated decision, execution, verification, and exception events.	Open control incident; prevent silent closure.	Hash-linked event chain and trace correlation.
I8 Idempotent effect	Idempotency key and target semantics prevent duplicate commitment, or weaker	Reject duplicate; compensate and escalate on partial	Idempotency record, target receipt, compensation status.

	guarantees are declared.	effect.	
--	--------------------------	---------	--

Note. The invariants are technology-neutral. Implementations may add stronger controls but should not weaken these predicates without documenting residual risk.

The governed state machine in Table 6 is constrained by the commit-time invariants in Table 7. I1 requires complete mediation; I2 enforces tier-specific outcomes; I3 requires evidential closure before review or authorization; I4 requires exact action/evidence/policy binding; I5 rejects stale state and replay; I6 enforces qualified independent authority; I7 requires lifecycle-complete event correlation; and I8 requires idempotent execution or an explicitly recorded weaker guarantee. Together, these invariants define what it means for a PAVE-CP implementation to authorize rather than merely observe an agent action.

Conditional approvals are represented as machine-enforceable constraints, not solely as prose. Examples include a maximum affected-row count, an approved recipient set, a monetary ceiling, an execution window, a report-period hash, or a required transaction mode. If the adapter cannot enforce a condition, the broker must not issue a token that suggests otherwise. The reviewer can instead reject and request a new, enforceable proposal.

Policy or evidence-service failure never silently becomes allow. Low-risk workflows may define a safe degraded state—for example, permitting T0 local metadata reads while disallowing egress and all state change. Higher tiers remain closed. Break-glass actions require a distinct policy namespace, named incident authority, short validity, separate credentials, explicit justification, and automatic post-event review. Availability exceptions are therefore visible control decisions rather than undocumented branches in application code.

5.7 Decision ledger and observability correlation

The PAVE ledger is event-sourced. Typical events include ACTION_PROPOSED, POLICY_DECISION, EVIDENCE_COLLECTED, EVIDENCE_VALIDATED, APPROVAL_REQUESTED, APPROVAL_GRANTED, APPROVAL_REJECTED, AUTHORIZATION_CONSUMED, EXECUTION_STARTED, EXECUTION_COMPLETED, POSTCONDITION_VERIFIED, COMPENSATION_STARTED, EXCEPTION_RAISED, and ACTION_CLOSED. Each event carries run, thread, tool-use, decision, trace, actor, timing, policy-version,

digest, outcome, and prior-event references.

The ledger is deliberately distinct from model-observability traces. A trace explains component activity and timing; the ledger records authoritative control decisions and their bindings. The two are correlated rather than conflated. Sensitive tool inputs and outputs can remain in a protected observability store while the PAVE event retains a content hash, classification, access pointer, and retention class. The resulting authorization receipt allows an auditor to reconstruct who authorized what, under which policy and evidence, without broadening routine access to the underlying data.

Logging policy must itself be governed. Secrets, credentials, personal data, and legally privileged material should not be copied into broad audit channels. Redaction and tokenization must preserve the fields needed to prove scope and integrity. Retention, legal hold, cross-border storage, access review, and deletion exceptions remain organizational controls beyond the controller core; the event schema exposes classification and retention fields so that those controls can be applied consistently.

6. Integration with Snowflake Cortex Agents

6.1 Integration boundary

Cortex Agents provides managed planning and tool orchestration over structured and unstructured enterprise data. Documented surfaces include streamed run events, tool-use objects with unique tool-use identifiers, permission options and client-supplied permission decisions for applicable executions, custom tools implemented as stored procedures or user-defined functions, immutable named agent versions and aliases, monitoring, and AI observability traces [33–38]. PAVE-CP uses only these public surfaces and does not depend on undisclosed planner state. Table 8 maps the control-plane obligations to these documented platform surfaces.

The preferred pattern is an application-side Cortex gateway that terminates the streamed agent response. When a response.tool_use item arrives, the gateway inspects the tool type, input, tool_use_id, client-side

execution indicator, and any permission options; constructs a PAVE action envelope; and calls the PDP. For a governed client-side tool, the adapter executes only after PAVE authorization. Where the upstream event exposes permission choices, the gateway translates the completed organizational decision into the compatible one-time `permission_decision` for the same `tool_use_id` [34].

A platform permission choice is not treated as the complete organizational approval record. The richer PAVE decision, evidence checklist, reviewer identities, conditions, digests, and lifecycle events remain in the external control plane. The platform decision is simply the protocol signal needed to continue or stop a particular upstream tool use. This separation avoids assigning evidentiary meaning to a binary user-interface choice that does not carry role, evidence, or policy bindings.

Not every managed tool path necessarily exposes a pre-execution interception point with the granularity

required by enterprise policy. Consequential operations should therefore be registered only as governed proxy procedures or functions. The proxy obtains PAVE authorization before invoking a lower-level business procedure, and the agent role has no direct privilege on the underlying write. Read-only analytical tools may remain directly available when their data, egress, and cumulative-use constraints are acceptable. This is the decisive trust-boundary assumption of the integration.

Publication must be modeled as a distinct action even when a report was drafted inside the agent conversation. Showing a draft to an internal analyst may be T1 or T2; writing an authoritative reporting table, sending to an external recipient, or submitting to a regulator is T4. The publication adapter therefore re-canonicalizes the final artifact, reconciliations, recipients, report period, and policy version, then obtains approvals for that exact publication object.

Table 8. Mapping PAVE-CP to documented Cortex Agents surfaces.

Cortex surface	Relevant documented signal	PAVE-CP use
Agent invocation	run/thread context and querying identity	Initialize control context and purpose; attach policy namespace.
response.tool_use	tool_use_id, tool name/type, inputs, client-side execution indicator, permission options where present	Create canonical action envelope; correlate one decision to one proposed tool use.
permission_decision	tool_use_id plus an allowed option such as one-time allow or deny	Transmit the final platform-level decision only after PAVE authorization.
Custom tool	stored procedure or UDF with configured execution context	Expose a governed proxy rather than direct high-impact business operation.
Agent versioning	immutable named version or alias	Include agent version in evidence and policy inputs; support controlled rollout/rollback.
Monitoring and AI observability	conversation, planning, tool, SQL, chart, input/output and trace spans	Correlate operational traces with PAVE decision and evidence records.
Snowflake access control	user default role, agent privileges, tool-object privileges	Provide least-privilege substrate; PAVE adds context-specific authorization rather than replacing RBAC.

Note. Mappings reflect documented public interfaces as of 13 July 2026. Availability and semantics can change; deployments should perform capability and conformance testing [33–38].

6.2 Gateway protocol

Algorithm 1. Platform-neutral Cortex gateway protocol (pseudocode).

```

on_tool_use(event):
    a = canonical_action_envelope(
        event, identity, purpose, agent_version
    )
    write(ACTION_PROPOSED, ids(a), hash(a))
    d = policy_decision_point.evaluate(a, context)
    e = evidence_service.collect_and_validate(a,
        d.obligations)

    if d.effect == "DENY" or not e.complete:
        ledger.close_denied(a, d, e)
        send_permission_decision(
            event.tool_use_id, "Deny", d.reason_codes
        )
        return

    approvals = []
    if d.effect == "REQUIRE_APPROVAL":
        approvals = approval_broker.obtain(
            packet=(a, d, e), roles=d.roles, count=d.count
        )

    z = execution_gate.authorize(a, d, e, approvals)
    if not z.ok:
        ledger.close_denied(a, d, e, z.reason)
        send_permission_decision(
            event.tool_use_id, "Deny", [z.reason]
        )
        return

    send_permission_decision(
        event.tool_use_id,
        compatible_allow_once(event)
    )
    actual = governed_adapter.execute(a, z)
    verify_postconditions_and_close(
        a, d, e, approvals, actual
    )

```

The ordering in Algorithm 1 is intentional. The gateway does not summon a reviewer before evidence is complete, because an incomplete packet cannot support meaningful scrutiny. It also does not emit the upstream allow signal until the execution gate has atomically validated and consumed the authorization. Production

implementations should make both the final permission signal and the governed executor idempotent so that stream retries, reconnects, or duplicate deliveries cannot repeat an effect.

Policy and agent versions are independent but jointly relevant. A stable agent version can be governed by a newly activated policy bundle; a newly committed agent version can operate under an unchanged policy. Both identifiers and digests are recorded. Cortex Agents supports mutable live versions, immutable named versions, aliases, promotion, and rollback [38]. Regulated workflows should target an immutable named version or controlled alias and should test policy and agent changes against historical action envelopes before promotion.

6.3 Data and privilege architecture

PAVE-CP complements Snowflake privileges rather than bypassing them. The agent and gateway operate with only the privileges needed to propose, read, and stage actions. High-impact business procedures require a separate executor identity available solely to the governed adapter after authorization. A common pattern is propose → stage → approve → commit: the agent creates a temporary object, the control plane approves its content digest, and a tightly scoped procedure promotes that exact object into the authoritative state.

The evidence service can remain close to governed data. Source queries can return query identifiers, counts, reconciliations, hashes, and lineage pointers without copying full restricted datasets into the reviewer interface. Reviewers receive masked examples or aggregate evidence according to their own access rights. Evidence minimization is a first-class requirement: the approval broker must not become an unintended exfiltration channel merely because a reviewer is authorized to decide.

Cortex monitoring and AI observability provide execution detail, including inputs, outputs, and intermediate steps [35,36]. PAVE-CP adds records that ordinary traces do not necessarily encode: the active policy decision, mandatory evidence checklist, role resolution, signed approvals, conditions, nonce consumption, and commit verification. The authoritative audit view joins both record sets by run, thread, tool-use, agent-version, and trace identifiers,

while enforcing separate access to unredacted evidence.

6.4 Deployment patterns and conformance boundary

Three deployment patterns are possible. In a native-interception pattern, the gateway pauses a client-side tool use, runs PAVE-CP, and returns the compatible platform decision. In a governed-proxy pattern, the agent can call only a proposal or staging tool, while a separately privileged adapter performs commitment. In a publication-control pattern, the generated artifact remains non-authoritative until a distinct publish action is authorized. The second and third patterns provide the strongest boundary because they remove direct write privilege from the agent identity.

Conformance should be tested at the executor, not inferred from interface behavior. A conformant deployment demonstrates that no governed effect can occur without the current authorization receipt; that all effect-bearing parameters are canonicalized; that evidence and tokens are fresh and bound; and that the target result is correlated with the authorization event. Platform traces and UI approvals are supporting evidence, not substitutes for these predicates.

7. Worked workflow scenarios

7.1 High-speed anomaly triage

Consider an agent investigating a divergence in a governed revenue pipeline. It reads metadata, queries a semantic view, compares source partitions, calculates a variance, and retrieves the incident-control procedure. These T0–T1 actions are read-only and contained, so they proceed automatically. Query identifiers, source windows, lineage, and the immutable agent version are captured as evidence without asking a reviewer to approve each analytical step.

The agent then proposes a non-authoritative incident case containing the anomaly summary and source links. This T2 action is reversible. If it has no restricted egress and targets an internal staging queue, policy may permit it after schema validation and duplicate-case checking. The control plane records the before/after representation, expected effect, and rollback method.

Quarantining a production partition is different. The T3 proposal includes the exact partition identifier, affected downstream reports, anomaly magnitude, source and comparison queries, dry-run object count, before/after

status, rollback procedure, and unresolved uncertainty. One qualified domain approver is required. The approval expires after 15 minutes and binds to that partition. Expanding the target from one partition to an entire schema changes the action digest and invalidates the token.

Drafting a management summary remains T2; publishing it to an authoritative executive channel may become T3, and sending it externally or including restricted records can become T4. The decomposition keeps investigation fast while making consequence-bearing transitions explicit. It also produces an audit record that explains not only that a quarantine or notification occurred, but which evidence and authority supported it.

7.2 Regulated reporting and authoritative publication

In a periodic regulated-reporting workflow, the agent retrieves approved source tables, performs calculations, and drafts explanatory narrative. Read and calculation steps are T1; storing a draft in a non-authoritative workspace is T2. The evidence bundle records report period, source-snapshot identifiers, query IDs, calculation procedure version, data-quality checks, and links to relevant control definitions.

Before the report can enter review, policy requires reconciliation to the authoritative ledger, schema and completeness validation, variance-threshold checks, and an explicit list of unresolved exceptions. An evidence failure does not generate an approval request. The agent receives a reason-coded remediation response and can collect missing evidence or produce a corrected proposal.

The final publication is T4. The canonical action includes the report-artifact hash, reporting entity, period, destination, recipient or filing endpoint, submission mode, and attachments. The evidence packet includes final reconciliations, control-owner attestations, material changes from the prior approved version, validation output, and any withdrawal procedure. A domain approver and an independent risk or compliance approver are required; requester, operator, and reviewers must satisfy configured separation-of-duties rules.

At commit, the adapter rechecks the artifact hash, destination, submission window, and policy version;

consumes both nonces; submits once using an idempotency key; captures the external receipt or authoritative object identifier; and verifies that the received artifact matches the approved digest. A later

narrative correction is a new transaction with new evidence and approval. Table 9 contrasts this staged reporting path with anomaly triage and shows where autonomy is intentionally preserved.

Table 9. Risk-adaptive control across the two reference workflows.

Workflow stage	Anomaly-triage example	Regulated-reporting example	Default tier	Control response
Observe / retrieve	Read pipeline metadata and incident procedure.	Read approved source tables and control definitions.	T0–T1	Automatic within data and egress constraints; capture query/source identifiers.
Analyze / reconcile	Compare partitions; calculate anomaly magnitude.	Calculate report values; run reconciliations and validations.	T1	Automatic; accumulate lineage, query IDs, uncertainty, and validation evidence.
Prepare / stage	Create non-authoritative incident case.	Store draft report in controlled workspace.	T2	Allow when reversible and contained; record diff, duplicate check, and rollback.
Material state change	Quarantine one production partition.	Mark controlled artifact ready for formal review.	T3	One qualified approval; dry run, affected-object limits, conditions, short expiry.
Authoritative publication	Notify external party or publish executive incident notice.	Submit or publish final regulated report.	T4	Two distinct role-qualified approvals, separation of duties, exact artifact/recipient binding.
Prohibited / exceptional	Unbounded destructive action or forbidden egress.	Submission to unapproved endpoint or without required authority.	T5	Deny normal path; separately governed break-glass process only where authorized.

Note. Tiers are defaults. Restricted egress, privilege change, irreversibility, prohibited targets, accumulated sequence risk, or other policy triggers can raise the effective tier.

7.3 Evidence-first reviewer experience

The reviewer interface should answer five questions without requiring transcript archaeology: What will happen? Why is it being proposed now? Which evidence and policy obligations support it? What could go wrong or remain uncertain? What exactly am I authorizing? The default view presents action scope, before/after diff, materiality, evidence-status indicators, policy reasons, expiry, rollback, and enforceable conditions; protected details are available through role-checked expansion.

A T4 reviewer can see whether all required attestations are present and whether another qualified reviewer has acted, but the interface should avoid social cues that

pressure conformity. Each decision is recorded with evidence version, timing, reason, and any condition. These data enable later analysis of workload and decision quality without equating rapid approval or a low rejection rate with good oversight.

Operational learning should feed policy governance. Repeated approval of a tightly constrained, reversible T2 action may justify automation under the same limits; recurring T2 incidents involving restricted egress may justify a hard trigger. A high approval rate with almost no detected defects may indicate strong evidence, low-value review, or fatigue. A high rejection rate may reflect poor proposals, missing evidence, or overly broad

workflow design. The control plane exposes these possibilities; human-factors study is still required to distinguish them.

8. Evaluation

8.1 Evaluation questions and baselines

The evaluation asks two different questions: does the reference controller enforce its declared structural invariants, and can the risk-adaptive architecture produce a plausible safety–speed advantage under transparent assumptions? It does not estimate real-world accident rates. The latent variable unsafe denotes a proposal that would violate the modeled safety objective if committed; high-impact unsafe restricts that measure to T3–T4 proposals. The unit of analysis is one proposed action envelope at one commit boundary.

Three policies are compared. B1, RBAC-only, denies T5 and lets all other actions execute under existing privileges

without standardized evidence. B2, blanket HITL, sends every T1–T4 action to a generic approval form. B3, PAVE-CP, requires one approval at T3 and two at T4, escalates lower tiers on configured hard or suspicious triggers, closes incomplete evidence, and applies tier-dependent verification before final commitment. T0 remains autonomous unless another rule denies it.

The hypotheses are directional. H1: PAVE-CP reduces modeled high-impact unsafe commits relative to B1 and B2. H2: PAVE-CP produces fewer review requests and lower median latency than B2. H3: every executed PAVE-CP action satisfies mandatory evidence obligations. H4: the controller rejects seeded violations of approval cardinality, freshness, exact-action binding, independence, evidence completeness, replay safety, and ledger integrity.

8.2 Synthetic workload and assumptions

Table 10. Base-case synthetic evaluation configuration.

Parameter	Base-case value or assumption
Scale	20 replications × 50,000 proposals = 1,000,000 actions; fixed master seed 20260713.
Workflow mix	65% anomaly triage; 35% regulated reporting.
Arrivals	Poisson arrivals at 90 proposals/hour; first-come, first-served reviewer queues.
Risk tiers	Triage: 45/30/17/6/1.5/0.5%; reporting: 25/24/23/19/7/2% for T0–T5.
Latent unsafe probability	Tier-dependent: 0.1%, 0.4%, 2.5%, 7%, 16%, and 45%.
Review capacity	Six reviewers; lognormal service; mean 4.2 minutes for blanket review and 2.8 minutes for structured PAVE review.
PAVE evidence	Tier-proportional assembly time; 0.8% independent evidence failure; all executed actions require complete mandatory fields.
Review detection	Unsafe-rejection probabilities are assumptions: 88.5% blanket, 95.5% PAVE first review, 93% PAVE second review.
Postcondition verification	Modeled unsafe-catch probabilities rise by tier from 20% at T0 to 94% at T4; T5 is denied.
Policy overhead	Synthetic lognormal overhead; results report modeled, not benchmarked, decision latency.

Note. The workload and service-time assumptions are synthetic and fully parameterized in the supplement.

The generator draws workflow type and tier, then samples restricted-data involvement, external egress, privilege change, latent unsafe state, an imperfect risk signal, and model confidence. Safe and unsafe proposals have overlapping signal distributions, so contextual escalation is neither perfectly sensitive nor perfectly specific. The simulation includes base reasoning and execution time, evidence assembly, policy-decision overhead, and role-specific reviewer queues.

Blanket review uses a generic form and a longer mean service time. PAVE review uses a structured evidence packet and a shorter service-time assumption, while T4 actions require a second sequential approval. Reviewer identities are represented as independent stages in the queue model; distinct identities and role coverage are

enforced separately in the executable authorization tests. This separation makes the queueing assumptions explicit rather than embedding them in the controller logic.

Every parameter is exposed in the supplementary code. The model is designed to reveal mechanisms and breakpoints, not to hide assumptions behind a headline number. A field study should replace probabilities and service-time distributions with organization-specific observations, calibrate hazard labels through expert adjudication, and evaluate reviewer behavior under an approved human-subjects protocol where applicable. Table 10 reports the base-case configuration; Table 11 defines how each outcome is operationalized and what it cannot establish.

Table 11. Evaluation constructs, operationalizations, and interpretation limits.

Construct / metric	Operationalization in the synthetic evaluation	Interpretation	Important limitation
Approval demand	Approval requests ÷ all proposals.	Human-review load created by a policy.	Does not measure cognitive effort or decision quality.
Bounded autonomy	Actions executed without human review ÷ all proposals.	Coverage of the autonomous fast path.	An autonomous action may still be denied or unsafe under a misspecified model.
Unsafe commits / 10,000	Latent unsafe proposals that execute, normalized by proposal count.	Comparative modeled hazard under common assumptions.	The latent label and hazard probabilities are synthetic, not observed incidents.
High-impact unsafe commits / 10,000	Unsafe committed T3–T4 proposals, normalized by proposal count.	Modeled failure at consequential boundaries.	Small counts create coarse replication intervals.
Latency	Proposal-to-terminal time; median, mean, and 95th percentile.	Separates fast autonomous path from governed tail.	Service-time distributions are assumptions; role topology is simplified.
Evidence completeness	Executed actions whose mandatory modeled evidence fields are complete.	Whether the controller enforces declared obligations.	Completeness does not prove evidence truth or legal sufficiency.
False rejection	Modeled safe proposals denied or rejected ÷ safe proposals.	Operational friction imposed on legitimate work.	Depends on the synthetic safety label and reviewer model.
Policy overhead	95th percentile of simulated PDP evaluation time.	Control-plane computational cost excluding human delay.	Not a benchmark of a specific production policy engine.

Note. The table separates construct definition from empirical claim. Synthetic metrics support mechanism comparison only within the stated workload model.

8.3 Comparative results

Table 12. Synthetic base-case safety and governance outcomes.

Policy	Approval %	Autonomy %	Unsafe commits /10k	High-impact unsafe /10k	False reject %
RBAC-only	0.00	98.97	192.17 [183.78–204.46]	129.79 [122.84–138.13]	0.59 [0.54–0.63]
Blanket HITL	60.97	38.00	25.39 [21.57–29.72]	14.92 [11.26–18.52]	2.27 [2.16–2.38]
PAVE-CP	16.54	81.78	7.13 [5.90–8.81]	0.36 [0.00–1.00]	1.67 [1.58–1.75]

Note. Brackets are 2.5th–97.5th percentile replication intervals across 20 replications of 50,000 proposals. Values are synthetic and should not be interpreted as production incident-rate estimates.

Table 13. Synthetic base-case operational outcomes.

Policy	Median latency (s)	P95 latency (s)	Mean latency (s)	Complete evidence %	PDP P95 (ms)
RBAC-only	1.99 [1.98–1.99]	3.43 [3.41–3.45]	2.10 [2.10–2.11]	63.46 [63.03–63.78]	0.00
Blanket HITL	162.25 [160.50–164.10]	463.87 [457.70–471.02]	167.45 [165.54–169.32]	79.56 [79.30–79.87]	7.31 [7.29–7.33]
PAVE-CP	2.34 [2.33–2.35]	237.10 [232.92–240.24]	35.70 [35.20–36.09]	100.00	22.51 [22.34–22.66]

Note. PDP = policy decision point. Complete evidence means that all modeled mandatory fields were present for executed actions; it does not establish truth or legal sufficiency.

Within the model, PAVE-CP reduced overall unsafe commits by 96.29% relative to RBAC-only and 71.92% relative to blanket HITL. For high-impact unsafe commits, the modeled reductions were 99.72% and 97.59%, respectively. These effects arise from the composition of deterministic triggers, evidence-supported review assumptions, dual approval at T4, evidence-failure closure, and postcondition verification. They are comparative outcomes of the stated simulation, not general empirical effectiveness estimates. Table 12 reports the safety and governance outcomes.

PAVE-CP requested approval for 16.54% of proposals—72.87% fewer requests than blanket HITL—and preserved 81.78% bounded autonomy, compared with 38.00% under blanket review. Median end-to-end latency was 2.34 seconds, approximately 69 times lower than the blanket median. These results support H2 in the base case because most PAVE-CP actions remain on the autonomous path. Operational outcomes are reported separately in Table 13 to avoid obscuring the safety measures in an over-wide table.

Tail behavior is more nuanced. The PAVE-CP 95th percentile was 237.10 seconds because the upper tail contains human-reviewed actions, although it remained below the blanket value of 463.87 seconds. The mean of 35.70 seconds was far above the median, reflecting a bimodal experience: rapid autonomous actions and slower governed actions. Figure 5 makes this dual-latency structure visible. A credible evaluation must report the tail and conditional review latency rather than relying on a single average.

Executed-action evidence completeness was 100% by construction and enforcement for PAVE-CP, compared with probabilistic completeness assumptions for the baselines. PAVE-CP also produced more false rejection than RBAC-only because it deliberately closed evidence failures and rejected some safe proposals during review. Bounded autonomy therefore introduces controlled friction; policy calibration must consider both safety misses and interruption of legitimate work.

Synthetic safety-speed-review trade-off under the base-case workload

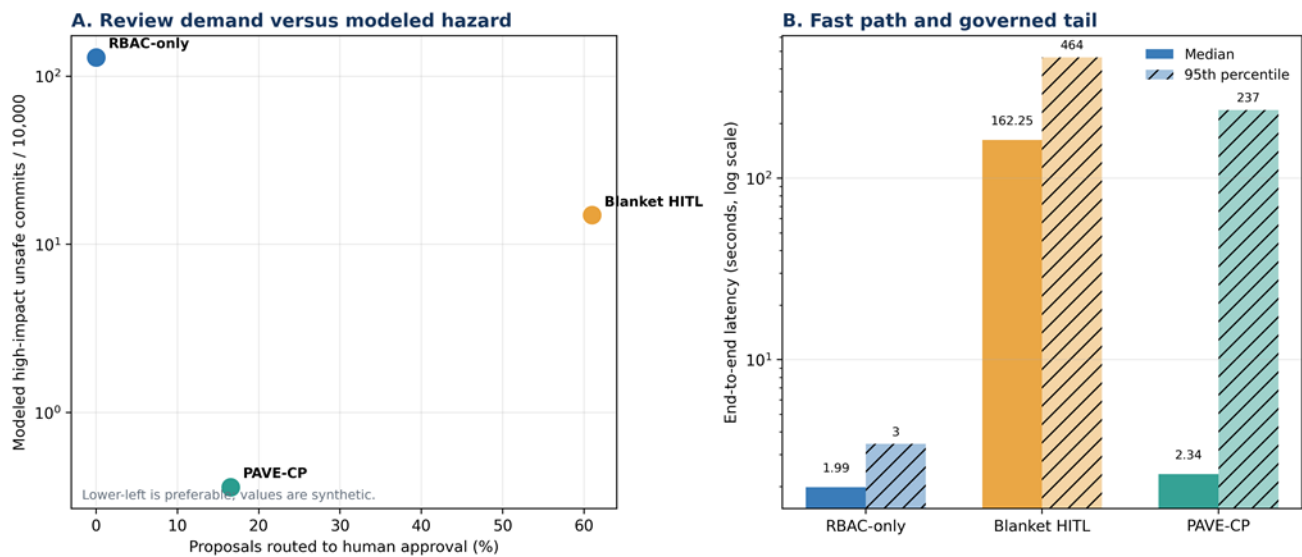


Figure 5. Synthetic safety–speed–review trade-off under the base-case workload.

Source: authors' simulation. The vertical axis in panel B uses a logarithmic scale.

8.4 Executable invariant and mutation tests

Table 14. Seeded control mutations detected by the reference artifact.

Mutation	Injected fault	Observed control response
Missing approval	Remove one token from a T4 dual-approval set	Rejected: approval cardinality
Stale approval	Set token expiry before commit time	Rejected: approval expired
Action drift	Change report parameter after approval	Rejected: action/evidence/approval binding mismatch
Role conflict	Reuse same approver identity for both attestations	Rejected: separation of duties
Missing evidence	Remove required dry-run result	Rejected: missing evidence
Replay	Reuse already consumed approval tokens	Rejected: nonce replay detected
Ledger tamper	Modify an earlier event payload	Detected: hash-chain verification failure

Note. Observed response refers to the executable artifact's expected rejection or integrity alert; these tests do not prove absence of other implementation defects.

The vectorized policy test generated 1,000,000 randomized states and found zero structural violations of the tier–effect mapping. The end-to-end test generated 20,000 action envelopes, evidence bundles, decisions, and valid approval sets; 19,790 authorizations and 210 denials

were expected, and no structural mismatch was observed. These results demonstrate consistency over the exercised state space, not a proof of all possible implementation behaviors.

All seven seeded mutations in Table 14 were detected. Action drift is especially important because it distinguishes approval workflow from transaction authorization. A system can correctly record that two people clicked “approve” and still be unsafe if the executor changes parameters afterward. PAVE-CP treats action, evidence, policy, reviewer authority, and commit state as one predicate checked immediately before effect.

The reference artifact is intentionally compact and dependency-light so that reviewers can inspect it. Production hardening would require typed service contracts, durable and replicated state, concurrency and fault-injection tests, key rotation, evidence-access authorization, signed policy distribution, multi-region replay prevention, secure time, independent penetration testing, and continuous conformance checks at the executor boundary.

8.5 Sensitivity to reviewer capacity and reporting mix

Sensitivity analysis varies available reviewers from three to ten and the regulated-reporting share from 15% to 65%. As the reporting share rises, both human-review designs route more consequential actions to reviewers, but blanket HITL generates substantially greater queue demand. With three reviewers and a 35% reporting share, the blanket queue is unstable under the modeled arrival

and service assumptions, producing a 95th-percentile delay of approximately 155,785 seconds. PAVE-CP remains near 237 seconds because its review arrival rate is much lower. Figure 6 shows the full reviewer-by-workload surface.

At six reviewers and a 35% reporting share, blanket and PAVE-CP 95th-percentile latencies are approximately 461 and 233 seconds. Adding blanket-review capacity beyond six reviewers reduces queueing but does not remove per-action review time or the cognitive burden of unnecessary requests. In the stable region, PAVE-CP is comparatively insensitive to reviewer count because the queue is lightly loaded; its reporting-share sensitivity mainly reflects a larger fraction of actions entering the governed path.

The sensitivity result is not an argument for minimizing reviewer headcount. T4 dual control, skill coverage, jurisdiction, leave, and conflict rules can make effective capacity lower than nominal staffing. Organizations should model reviewer pools by role and time zone, define service objectives by tier, and monitor queue utilization. The value of selective escalation is that scarce attention can be reserved for decisions where human authority and domain judgment materially change the safety case.

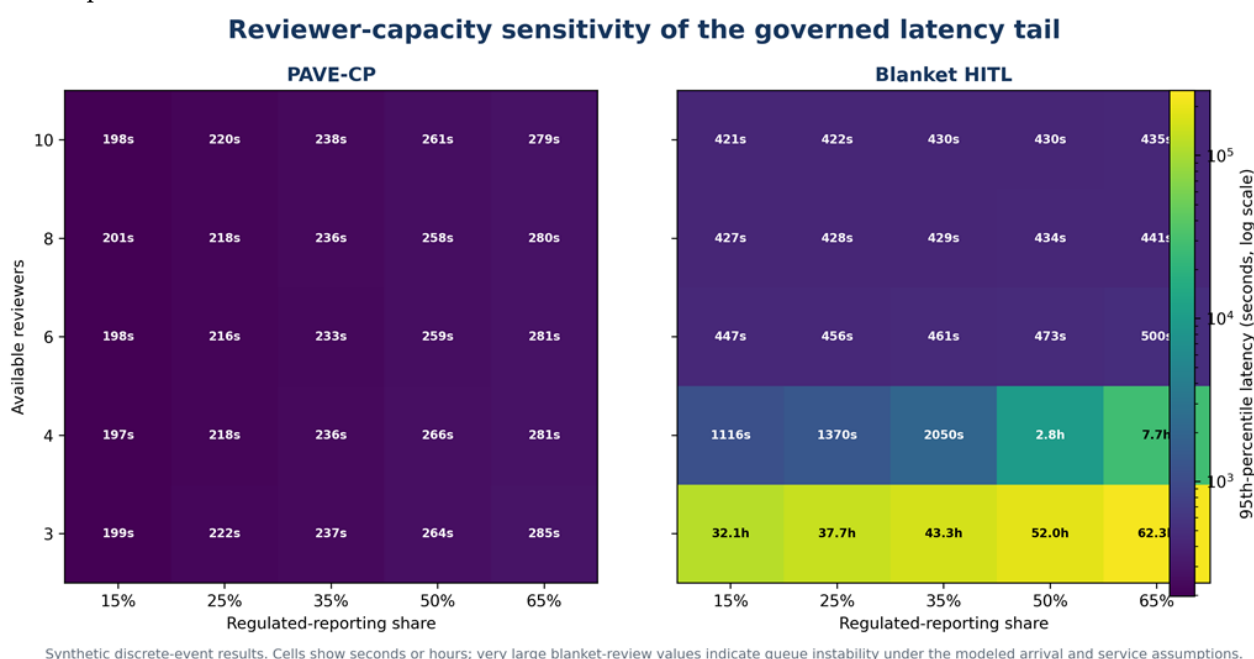


Figure 6. Sensitivity of 95th-percentile latency to reviewer capacity and regulated-reporting share.

Source: authors’ simulation. Very large blanket-review cells indicate queue instability under the modeled arrival and service assumptions.

8.6 Validity, reproducibility, and interpretation

The evaluation combines deterministic invariants,

mutation analysis, repeated simulation, and sensitivity analysis, which strengthens internal coherence but does not substitute for field evidence. Construct validity

depends on the definitions in Table 11; internal validity depends on the executable rules and seeded assumptions; external validity is intentionally limited. The supplement exposes all parameters and result files, allowing alternative hazard models, reviewer service distributions, and workflow mixes to be tested without changing the core controller.

8.7 Evaluation summary

H1–H4 are supported within the boundaries of the artifact tests and synthetic model. The controller enforced the selected invariants and detected the seeded mutations. The simulation demonstrated a coherent mechanism by which evidence-first, risk-adaptive control can reduce review demand and modeled unsafe commitment while preserving low median latency. It also confirmed the residual operational fact that every action requiring a person inherits human service time and queue risk; PAVE-CP concentrates that latency rather than eliminating it.

9. Discussion

9.1 Theoretical contribution

The central theoretical move is to replace “human-in-the-loop” as a workflow label with human authorization as a verifiable commit predicate. Human presence is not the control objective. The objective is that a qualified person receives sufficient evidence, decides within a defined scope, and that the exact decision is enforced when a consequence is created. This framing supports comparison across platforms and domains because it specifies observable predicates rather than interface presence.

A second contribution is evidential closure. An approval request is ready only when every policy-mandated, externally inspectable item is present, valid, attributable, and fresh. This condition prevents the system from shifting evidence discovery onto the reviewer and creates a machine-testable boundary between proposal and approval. It also clarifies why a fluent rationale is useful context but not a substitute for source, lineage, validation, and effect evidence.

A third contribution is cross-layer authorization binding. Policy engines bind decisions to inputs; approval systems bind people to forms; execution systems bind credentials to calls; observability systems bind events to traces. PAVE-CP composes those bindings through

common identifiers, canonical representations, and digests. The composition addresses policy drift, approval substitution, replay, and unlogged parameter change—failure modes that no component can prevent in isolation.

The fourth contribution is a dual-latency model of governed autonomy. A well-designed agent system has a fast path for low-risk work and a deliberate path for consequential work. Policy quality determines which path an action enters; evidence quality, reviewer capacity, and role topology determine the governed tail. Evaluation should therefore report autonomy coverage, approval demand, and tier-conditional latency rather than only aggregate response time.

9.2 Practical implications for enterprise architecture

Enterprises should treat the control plane as shared infrastructure rather than embedding bespoke approval logic in every agent. Shared schemas and policy decisions enable consistent enforcement across anomaly, finance, security, operations, and reporting workflows. Domain teams still own thresholds, evidence obligations, and reviewer roles, but publish them into a governed policy lifecycle. Platform team’s own availability, identity integration, cryptographic binding, replay safety, observability, and the executor boundary.

The first deployment priority is a consequential-tool inventory. Each tool is classified by potential effect, reversibility, data boundary, and authoritative status; direct high-impact tools are replaced by propose/stage/commit interfaces. The second priority is reusable evidence adapters for common controls such as lineage, dry runs, affected-object limits, report reconciliation, recipient validation, and rollback. The third is reviewer operations: qualified pools, delegation, service objectives, escalation, and after-hours coverage.

Policy-as-code must be governed like production software. Rules require ownership, peer review, unit tests, counterfactual simulation against historical envelopes, versioned bundles, signatures, staged deployment, canaries, metrics, and rollback. A policy digest in an approval token is useful only if the corresponding bundle is authenticated, retrievable, and linked to its deployment decision. Emergency policy changes need the same evidentiary discipline, albeit through an accelerated and separately authorized route.

The ledger supports control testing and targeted audit sampling. Auditors can query all T4 actions, approvals near expiry, repeated denials, evidence overrides, failed postconditions, policy changes, or actions whose actual effect approached an approved limit. The objective is not indiscriminate surveillance; it is selective reconstruction of control performance under access restrictions appropriate to the sensitivity of the evidence.

9.3 Human factors and approval fatigue

Approval fatigue is a property of the system, not a defect in individual reviewers. It emerges when low-value requests are over-routed, packets are difficult to compare, consequences are unclear, or organizational incentives reward throughput over scrutiny [32,40,43,45]. PAVE-CP reduces volume through risk tiers and improves salience through structured evidence and diffs. It cannot create domain expertise, psychological safety for rejection, or adequate staffing.

Organizations should monitor conditional measures

such as time to first view, deliberation time after view, reason-code distribution, evidence expansion, changed decisions, post-approval exceptions, and repeated approvals of identical constrained patterns. Faster approval is not inherently better, and low rejection is not inherently evidence of success. Blinded quality review and adversarial control tests can help determine whether reviewers detect seeded defects rather than merely process volume.

Automation can expand when evidence and constraints make an action mechanically safe. Repeated approval of a reversible T2 update with a one-record limit may justify an automatic rule under the same conditions. Conversely, recurring T2 incidents involving restricted egress may justify mandatory escalation. The learning loop preserves human attention by translating stable decisions into policy while keeping novel, ambiguous, or high-impact decisions visible.

9.4 Security and failure analysis

Table 15. Threat model, mitigations, and residual risk.

Threat/failure	Potential consequence	PAVE mitigation and residual risk
Prompt injection or untrusted tool data	Agent proposes malicious or unrelated action	External PEP, purpose/target policy, restricted-egress trigger, exact-action review; still requires content security and least privilege.
Hallucinated parameters	Wrong identifier, recipient, date, or magnitude	Typed action schema, target resolution, dry run, diff, affected-object bounds, postconditions.
Over-privileged agent	Directly bypasses gateway	Remove direct write privilege; expose governed proxy only; network/service identity controls.
Stale approval / TOCTOU	State changes after review	Short TTL, precondition hash, revalidation at commit, new proposal on drift.
Replay or retry duplication	Same token or action executes twice	Atomic nonce consumption, idempotency key, durable execution receipt.
Compromised reviewer	Improper but correctly signed approval	Role limits, dual control, anomaly detection, rotation, downstream constraints; cannot be solved by cryptography alone.
Ledger tampering	Decision history altered	Hash chain, signatures, protected append store, independent retention/anchoring and access monitoring.
Control-plane outage	Pressure to bypass control	Tier-specific degraded policy, fail closed for consequential actions, explicit break-glass with retrospective review.
Policy defect	Incorrect classification or evidence obligations	Policy tests, staged rollout, decision simulation, bundle signing, rollback, monitoring and independent assurance.

Note. Residual risk assumes the PEP, executor, identity, key, time, and retention services are independently secured and monitored.

Table 15 summarizes the principal threats, mitigations, and residual risks. The trusted computing base remains substantial: the PEP, identity provider, policy distribution, evidence adapters, key service, nonce and idempotency stores, executor, retention system, and time source. PAVE-CP reduces the need to trust the language model and reviewer interface, but a compromised PEP or executor can bypass logical checks. Service isolation, least-privilege credentials, independent monitoring, attested builds where available, and periodic end-to-end control tests remain necessary [16,17,41,47].

9.5 Relationship to compliance

PAVE-CP provides technical support for accountability, traceability, human oversight, and control evidence. It can demonstrate that a defined policy was applied and that qualified reviewers authorized an exact action on specified evidence. It does not determine

whether a workflow is legally high-risk, whether a role is legally competent, whether the evidence satisfies a sector-specific obligation, or whether the use is fair, necessary, and proportionate. Those judgments require legal, risk, compliance, domain, data-protection, security, and worker participation.

The artifact should therefore be embedded in an AI management system rather than marketed as a compliance product. Governance owners define risk criteria and accountability; domain control owners define evidence and reviewer competence; security owns the trust boundary; data governance owns classification and lineage; and internal audit independently tests design and operating effectiveness. Table 16 provides an illustrative, non-normative mapping from governance concerns to PAVE-CP mechanisms and the evidence they can produce.

Table 16. Illustrative governance alignment and evidence outputs.

Governance source / concern	PAVE-CP mechanisms	Control evidence produced	Boundary of the mapping
NIST AI RMF / GenAI Profile [4,5]	Contextual risk tiering; named owners; policy lifecycle; human-oversight rules; monitoring.	Policy decision, owner/role resolution, approval packet, verification and exception metrics.	Supports risk-management implementation; does not determine acceptable risk or organizational accountability.
EU AI Act high-risk oversight and logging [6]	Qualified intervention points; event retention; exact-action authorization; override/stop paths.	Reviewer identity and competence attributes, decision reasons, action/evidence bindings, execution receipt.	Illustrative technical support only; legal classification and conformity assessment remain external.
ISO/IEC 42001 AI management system [7]	Governed policy and agent versions; change control; responsibilities; operating-effectiveness evidence.	Version history, deployment approvals, decision samples, control exceptions, review metrics.	Does not constitute certification or satisfy all management-system clauses.
ISO/IEC 23894 and 5338 risk/lifecycle processes [8,9]	Risk treatment at action boundaries; staged deployment; postcondition and incident feedback loops.	Tier rationale, validation evidence, rollback/compensation, lifecycle event chain.	Domain risk identification and lifecycle governance still require organizational processes.
NIST SP 800-53 / 800-207 and ABAC [16,17,48]	Least privilege; continuous authorization; complete mediation; separation of duties; fail-closed execution.	Identity and role checks, policy digest, nonce state, executor authorization, bypass alerts.	Secure infrastructure and independent control assessment remain necessary.
OWASP agentic-	External PEP; governed	Blocked tool misuse,	Does not eliminate model

risk concerns [47]	tools; egress triggers; exact binding; sequence-ready event data.	privilege or target violations, prompt-injection-related escalations, correlated traces.	compromise, supply-chain risk, or cumulative multi-agent hazards.
---------------------------	---	--	---

Note. This crosswalk is illustrative and non-normative. It does not constitute legal advice, certification, or a statement of conformity.

9.6 Relationship to adjacent runtime-authorization work

AEGIS and AARM share the important premise that tool execution is the stable security boundary and should be mediated before side effects [31,44]. PAVE-CP extends that premise for regulated organizational use by making evidence obligations, reviewer roles and cardinality, exact evidence/policy binding, separation of duties, conditional constraints, postcondition verification, and managed-platform correlation part of the normative transaction. These systems are therefore complementary rather than mutually exclusive: a runtime firewall can supply interception and content scanning, while PAVE-CP supplies the evidence-backed authorization contract.

Delegation and permission research focus on who grants an agent authority, how scope is communicated, and how users participate in decisions [43,45,46]. PAVE-CP treats those questions as inputs to a broader organizational commit predicate. The relevant reviewer may be a domain owner, risk officer, or other qualified role rather than the end user; multiple independent reviewers may be mandatory; and approval is withheld until evidence closure. This distinction is central to regulated reporting and other workflows in which authority is institutional, not merely personal.

10. Limitations and future research

The principal limitation is empirical. Comparative outcomes come from a stylized simulation with assumed tier distributions, hazard probabilities, reviewer sensitivities, service times, and verification effectiveness. The model is transparent and reproducible, but its numerical values cannot be generalized to production. A next study should instrument a shadow-mode deployment, collect action envelopes and reviewer timing without enabling consequential automation, calibrate the model, and then conduct a controlled field trial.

No human participants were included. The study therefore cannot establish whether the proposed packet improves comprehension, calibration, or defect detection,

or whether dual approval introduces anchoring and diffusion of responsibility. Future work should compare packet designs in realistic tasks, measure workload and decision quality, and test whether reviewers detect action drift, misleading rationale, and incomplete evidence under time pressure.

Risk-tier assignment is policy-dependent and may encode organizational blind spots. A learned risk score can be manipulated or poorly calibrated. PAVE-CP mitigates this risk by preserving deterministic hard triggers and preventing probabilistic signals from lowering mandatory controls, but policy errors remain possible. Formal policy analysis, historical counterfactual evaluation, coverage testing, independent review, and sequence-aware budgets are needed.

The reference controller demonstrates logical binding, not production cryptography. HMAC does not provide independent non-repudiation when one service holds the shared key; an in-memory nonce example is not multi-region durable; and a hash chain detects only some forms of mutation. Production research should evaluate managed asymmetric signatures, transparency logs, trusted timestamping, external anchoring, truncation detection, and privacy-preserving audit proofs [25,41].

The Cortex mapping is based on public documentation current at the manuscript date. Platform interfaces, tool types, permission semantics, preview status, and monitoring schemas can change. Implementations should perform capability discovery and conformance tests, avoid undocumented behavior, and preserve governed proxy tools and least privilege when native interception does not meet the required control boundary.

Postcondition verification is domain-specific. Database row counts, report hashes, and message receipts can be mechanically checked; social, financial, clinical, or legal consequences may not be. The artifact

can prove that a declared verifier ran and record its result, but it cannot guarantee that the verifier captures every relevant harm. Domain libraries of preconditions, invariants, and compensating actions are an important research direction.

Auditability and evidence minimization can conflict. Rich packets improve review and reconstruction but increase privacy, confidentiality, and retention exposure. Future work should study selective disclosure, role-specific packet views, cryptographic commitments, zero-knowledge attestations for narrow control predicates, and policies that distinguish evidence needed at approval from evidence retained for audit.

Finally, the design assumes that consequential boundaries can be identified. Agents can create cumulative risk through repeated low-tier queries, incremental data transfer, coordinated changes, or multi-agent delegation. Future policy models should incorporate budgets, temporal aggregation, graph-level plans, recipient histories, and cross-agent correlation. The event model supplies data for such sequence-aware policy, but the present evaluation focuses primarily on individual commit boundaries.

11. Conclusion

Agentic speed and regulated control are compatible when proposal is separated from authorization. PAVE-CP provides that separation through an external, policy-driven commit protocol. It automates low-risk investigation and drafting, escalates consequential actions with tier-appropriate evidence, binds human decisions to exact action/evidence/policy digests, enforces freshness and separation of duties, verifies effects, and records a reconstructable lifecycle.

The design changes the governing question from “Did a person see the agent output?” to “Was this exact consequence authorized by the required authority on complete and fresh evidence under the active policy, and can the organization prove what executed?” That formulation is technically stronger, operationally measurable, and better aligned with regulated workflows.

The executable artifact and synthetic evaluation show a coherent mechanism by which selective escalation can preserve a rapid median path while concentrating review on high-impact boundaries. They also expose unresolved costs: approval-bound tail latency, false rejection, policy

maintenance, evidence privacy, and the size of the trusted computing base. These costs should be managed explicitly through human-factors study, policy lifecycle controls, capacity planning, and independent assurance.

For Cortex Agents, the practical architecture combines documented tool-use and permission interfaces, governed custom-tool proxies, least-privilege roles, immutable agent versions, and platform observability with an external PAVE authorization record. The result does not certify compliance. It provides a standardized, testable foundation on which accountable agentic workflows can be implemented and audited.

Declarations

Ethics statement. No human participants, personal data, production records, or live regulated decisions were used. All quantitative outcomes were generated from synthetic data and explicit assumptions.

Data and code availability. The accompanying reproducibility supplement contains the reference controller, JSON schemas, illustrative Rego policy, invariant and mutation tests, simulation and sensitivity code, machine-readable results, and all manuscript figures.

Funding. No external funding is declared for this anonymized manuscript.

Conflicts of interest. No conflicts of interest are declared.

Author contributions. Withheld for double-anonymous review.

Compliance statement. PAVE-CP is a technical control design. Its deployment does not by itself establish conformity with any law, regulation, standard, contractual obligation, or sector-specific control framework.

Appendix A. Normative PAVE-CP authorization schema

A conformant implementation should expose at least the logical objects in Table A1. Field names and transport formats may differ, but the binding, freshness, cardinality, and lifecycle semantics should remain interoperable.

Table A1. Minimum interoperable control objects.

Object	Required logical content
ActionEnvelope	identity; run/thread/tool-use IDs; agent/version; purpose; action/tool; inputs; targets; classifications; egress; privilege change; reversibility; impact; uncertainty; timestamps; idempotency key; action hash
PolicyDecision	decision ID; effect; tier; evidence obligations; approval count; required roles; reason codes; postconditions; policy version/hash; validity interval
EvidenceBundle	action hash; source and lineage IDs; citations; parameter diff; dry run; validations; expected effect; rollback; uncertainty; trace context; freshness; content hashes; evidence hash
ApprovalToken	token/decision/tool-use IDs; action/evidence/policy hashes; approver identity/roles; choice; enforceable conditions; issue/expiry; nonce; signature
AuthorizationReceipt	authorization ID; consumed tokens/nonces; validated hashes; executor identity; idempotency key; authorization time; permitted constraints
ExecutionRecord	actual parameters; start/end; affected objects; external receipt; before/after hashes; postcondition results; compensation status; trace context
AuditEvent	event ID/type; actor; correlated IDs; timestamp; outcome/reasons; content references/hashes; classification/retention; previous-event hash; signature

Note. Logical object names are normative for this manuscript only; interoperable implementations may use different field names and transports while preserving the same semantics

Appendix B. Illustrative policy decision**Listing B1. Simplified policy-as-code pattern; the full example is in the supplement.**

```

package pave.control
import rego.v1

default decision := {
  "effect": "DENY", "tier": 5, "approval_count": 0,
  "required_roles": [], "evidence_obligations": [],
  "reason_codes": ["FAIL_CLOSED"]
}

restricted_data if {
  some c in input.data_classes
  c in {"restricted", "confidential", "regulated", "personal"}
}

hard_trigger if input.privilege_change
hard_trigger if input.external_egress; restricted_data
hard_trigger if input.action_type in {
  "publish_report", "submit_regulatory_report"
}

effective_tier := max([input.estimated_impact, irreversible_floor])
irreversible_floor := 3 if not input.reversible
irreversible_floor := 0 if input.reversible

base_obligations := {
  "identity", "action_digest", "policy_digest", "trace_context"
}
tier4_obligations := {
  "dual_attestation", "separation_of_duties"
} if effective_tier >= 4 else set()
obligations := base_obligations | tier4_obligations

decision := {
  "effect": "REQUIRE_APPROVAL",
  "tier": effective_tier,
  "approval_count": 2,
  "required_roles": ["domain_approver", "risk_approver"],
  "evidence_obligations": sort(obligations),
  "reason_codes": ["HIGH_IMPACT", "DUAL_APPROVAL"]
} if effective_tier == 4

```

The listing is illustrative rather than a complete production policy. A production bundle should define schemas, default-deny behavior, rule precedence, conflict resolution, policy-data provenance, test cases, reason codes, validity periods, obligation composition, and signed distribution. Evaluation errors must produce a

controlled denial or tier-specific degraded decision, never an implicit allow.

Appendix C. Deployment and assurance checklist

1. Inventory every agent tool and identify the actual effect boundary, not only the displayed tool name.

2. Remove direct agent privileges to consequential executors and expose governed propose/stage/commit interfaces.
3. Define T0–T5 classifications, hard triggers, prohibited actions, evidence obligations, approver roles, and separation rules.
4. Implement canonical action schemas that expose all effect-bearing parameters and targets.
5. Protect policy bundles, evidence artifacts, signing keys, nonce/idempotency stores, trusted time, and audit retention.
6. Test missing evidence, policy drift, action substitution, stale approval, replay, duplicate delivery, partial execution, and compensation.
7. Run the control plane in shadow mode; compare decisions with existing controls; calibrate tiers and reviewer capacity.
8. Measure approval demand and quality by tier, not only aggregate response time or click-through rate.
9. Conduct domain, legal, data-protection, security, worker, and human-factors review before enabling consequential automation.
10. Maintain an explicit compliance mapping and independent operating-effectiveness testing; do not equate technical logging with legal conformity.

References

1. Hevner, A.R., March, S.T., Park, J. and Ram, S. (2004), "Design science in information systems research", *MIS Quarterly*, Vol. 28 No. 1, pp. 75–105. <https://doi.org/10.2307/25148625>.
2. Peffers, K., Tuunanen, T., Rothenberger, M.A. and Chatterjee, S. (2007), "A design science research methodology for information systems research", *Journal of Management Information Systems*, Vol. 24 No. 3, pp. 45–77. <https://doi.org/10.2753/MIS0742-1222240302>.
3. Gregor, S. and Hevner, A.R. (2013), "Positioning and presenting design science research for maximum impact", *MIS Quarterly*, Vol. 37 No. 2, pp. 337–355. <https://doi.org/10.25300/MISQ/2013/37.2.01>.
4. National Institute of Standards and Technology (2023), *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, Gaithersburg, MD. <https://doi.org/10.6028/NIST.AI.100-1>.
5. National Institute of Standards and Technology (2024), *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*, NIST AI 600-1, Gaithersburg, MD. <https://doi.org/10.6028/NIST.AI.600-1>.
6. European Parliament and Council of the European Union (2024), *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence*, *Official Journal of the European Union*, 12 July 2024.
7. International Organization for Standardization and International Electrotechnical Commission (2023), *ISO/IEC 42001:2023, Information technology—Artificial intelligence—Management system*.
8. International Organization for Standardization and International Electrotechnical Commission (2023), *ISO/IEC 23894:2023, Information technology—Artificial intelligence—Guidance on risk management*.
9. International Organization for Standardization and International Electrotechnical Commission (2023), *ISO/IEC 5338:2023, Information technology—Artificial intelligence—AI system life cycle processes*.
10. Parasuraman, R., Sheridan, T.B. and Wickens, C.D. (2000), "A model for types and levels of human interaction with automation", *IEEE Transactions on Systems, Man, and Cybernetics—Part A*, Vol. 30 No. 3, pp. 286–297. <https://doi.org/10.1109/3468.844354>.
11. Bainbridge, L. (1983), "Ironies of automation", *Automatica*, Vol. 19 No. 6, pp. 775–779. [https://doi.org/10.1016/0005-1098\(83\)90046-8](https://doi.org/10.1016/0005-1098(83)90046-8).
12. Goddard, K., Roudsari, A. and Wyatt, J.C. (2012), "Automation bias: a systematic review of frequency, effect mediators, and mitigators", *Journal of the American Medical Informatics Association*, Vol. 19 No. 1, pp. 121–127. <https://doi.org/10.1136/amiajnl-2011-000089>.
13. Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P.N., Inkpen, K., Teevan, J., Kikin-Gil, R. and Horvitz, E. (2019), "Guidelines for human–AI interaction", *Proceedings of CHI 2019*, Paper 3. <https://doi.org/10.1145/3290605.3300233>.
14. Shneiderman, B. (2020), "Human-centered artificial intelligence: reliable, safe and trustworthy", *International Journal of Human–Computer Interaction*, Vol. 36 No. 6, pp. 495–504. <https://doi.org/10.1080/10447318.2020.1741118>.
15. [15] Saltzer, J.H. and Schroeder, M.D. (1975), "The protection of information in computer systems", *Proceedings of the IEEE*, Vol. 63 No. 9, pp. 1278–1308. <https://doi.org/10.1109/PROC.1975.9939>.

16. [16] Rose, S., Borchert, O., Mitchell, S. and Connelly, S. (2020), Zero Trust Architecture, NIST Special Publication 800-207. <https://doi.org/10.6028/NIST.SP.800-207>.
17. Joint Task Force (2020), Security and Privacy Controls for Information Systems and Organizations, NIST Special Publication 800-53 Revision 5. <https://doi.org/10.6028/NIST.SP.800-53r5>.
18. OASIS (2013), eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard, 22 January 2013.
19. Open Policy Agent (2026), "Policy language and policy-as-code documentation", available at: <https://openpolicyagent.org/docs/policy-language> (accessed 13 July 2026).
20. [20] Open Policy Agent (2026), "Decision logs", available at: <https://openpolicyagent.org/docs/management-decision-logs> (accessed 13 July 2026).
21. Rundgren, A., Jordan, B. and Erdtman, S. (2020), JSON Canonicalization Scheme (JCS), RFC 8785. <https://doi.org/10.17487/RFC8785>.
22. World Wide Web Consortium (2013), PROV-O: The PROV Ontology, W3C Recommendation, 30 April 2013, available at: <https://www.w3.org/TR/prov-o/>.
23. World Wide Web Consortium (2021), Trace Context, W3C Recommendation, 23 November 2021, available at: <https://www.w3.org/TR/trace-context/>.
24. OpenTelemetry Authors (2026), OpenTelemetry Specification and Documentation, available at: <https://opentelemetry.io/docs/> (accessed 13 July 2026).
25. Haber, S. and Stornetta, W.S. (1991), "How to time-stamp a digital document", Journal of Cryptology, Vol. 3, pp. 99–111. <https://doi.org/10.1007/BF00196791>.
26. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. and Cao, Y. (2023), "ReAct: synergizing reasoning and acting in language models", 11th International Conference on Learning Representations.
27. Liu, X. et al. (2024), "AgentBench: evaluating LLMs as agents", 12th International Conference on Learning Representations.
28. Ruan, Y., Dong, H., Wang, A., Pitis, S., Zhou, Y., Ba, J., Dubois, Y., Maddison, C.J. and Hashimoto, T. (2024), "Identifying the risks of LM agents with an LM-emulated sandbox", 12th International Conference on Learning Representations.
29. Debenedetti, E., Zhang, J., Balunovic, M., Beurer-Kellner, L., Fischer, M. and Tramèr, F. (2024), "AgentDojo: a dynamic environment to evaluate prompt injection attacks and defenses for LLM agents", Advances in Neural Information Processing Systems, Vol. 37.
30. Turpin, M., Michael, J., Perez, E. and Bowman, S.R. (2023), "Language models do not always say what they think: unfaithful explanations in chain-of-thought prompting", Advances in Neural Information Processing Systems, Vol. 36.
31. Yuan, A., Su, Z. and Zhao, Y. (2026), "AEGIS: no tool call left unchecked—a pre-execution firewall and audit layer for AI agents", arXiv:2603.12621.
32. Wang, P., Li, Y. and Tian, Y. (2026), "Reframing LLM agent security as an agent–human interaction problem", arXiv:2605.24309.
33. Snowflake Inc. (2026), "Cortex Agents", Snowflake Documentation, available at: <https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents> (accessed 13 July 2026).
34. Snowflake Inc. (2026), "Cortex Agents Run API", Snowflake Documentation, available at: <https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents-run> (accessed 13 July 2026).
35. Snowflake Inc. (2026), "Monitor Cortex Agent requests", Snowflake Documentation, available at: <https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents-monitor> (accessed 13 July 2026).
36. Snowflake Inc. (2026), "AI Observability in Snowflake Cortex", Snowflake Documentation, available at: <https://docs.snowflake.com/en/user-guide/snowflake-cortex/ai-observability> (accessed 13 July 2026).
37. Snowflake Inc. (2026), "Create and manage agents", Snowflake Documentation, available at: <https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents-manage> (accessed 13 July 2026).
38. Snowflake Inc. (2026), "Cortex Agent versioning", Snowflake Documentation, available at: <https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents-versioning> (accessed 13 July 2026).
39. Lee, J.D. and See, K.A. (2004), "Trust in automation: designing for appropriate reliance", Human Factors, Vol. 46 No. 1, pp. 50–80. https://doi.org/10.1518/hfes.46.1.50_30392.
40. Parasuraman, R. and Riley, V. (1997), "Humans and automation: use, misuse, disuse, abuse", Human Factors, Vol. 39 No. 2, pp. 230–253. <https://doi.org/10.1518/001872097778543886>.

41. Schneier, B. and Kelsey, J. (1999), "Secure audit logs to support computer forensics", *ACM Transactions on Information and System Security*, Vol. 2 No. 2, pp. 159–176. <https://doi.org/10.1145/317087.317089>.
42. Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H. and Zhang, Y. (2025), "Agent Security Bench (ASB): formalizing and benchmarking attacks and defenses in LLM-based agents", *13th International Conference on Learning Representations*.
43. Brigham, N.G., Bagdasarian, E., Kohno, T. and Roesner, F. (2026), "Janus: a playground for user-involved agentic permission management", *arXiv:2607.01510*.
44. Errico, H. (2026), "Autonomous Action Runtime Management (AARM): a system specification for securing AI-driven actions at runtime", *arXiv:2602.09433*.
45. Zhang, Y.E. and Wang, G. (2026), "Towards human-centered agent authorization: a landscape analysis of commercial AI agents", *Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems*, Article 684, pp. 1–10. <https://doi.org/10.1145/3772363.3798851>.
46. South, T., Marro, S., Hardjono, T., Mahari, R., Whitney, C.D., Greenwood, D., Chan, A. and Pentland, A. (2025), "Authenticated delegation and authorized AI agents", *arXiv:2501.09674*.
47. OWASP GenAI Security Project (2025), "OWASP Top 10 for Agentic Applications for 2026", available at: <https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/> (accessed 13 July 2026).
48. Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R. and Scarfone, K. (2014), *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*, NIST Special Publication 800-162. <https://doi.org/10.6028/NIST.SP.800-162>.