OPENKEDGE / TECHNICAL WHITEPAPER

The Agentic Control Plane

An Evidence-to-Authority Architecture for Consequential AI Agents

Separating generation, judgment, authority, and execution

Core principle

An agent's proposal describes a possible action; a model's judgment is evidence about it. Neither is, by itself, authority to mutate protected state.

Architecture and protocol specification

Version 0.3 | 28 September 2026 | Design proposal

Jun He and Deying Yu | OpenKedge.io

Executive summary

AI agents can propose consequential operations against cloud infrastructure, enterprise applications, IAM permissions, databases, and financial systems. Many current architectures allow the same reasoning mechanism that generates or evaluates a tool call to directly trigger its execution. A model judgment, however accurate or confident, is merely evidence about an action; it is not authority to mutate protected state. The central architectural question is: *under what verifiable conditions does evidence become executable authority for this exact mutation, at this time?*

This whitepaper proposes an **Agentic Control Plane** (ACP) founded on an independently enforceable **Evidence-to-Authority boundary**. An agent submits a canonical proposal; decision providers return structured semantic, deterministic, or human evidence; and an assurance service evaluates obligations conditioned on the action's risk and consequences. Only successful admission permits an independent issuer to mint a narrowly scoped execution grant. At the mutation boundary, an execution gateway independently validates the grant, policy currency, and commit-time state preconditions. Authoritative outcome evidence then cryptographically links the complete authorization chain to the observed post-execution state.

Generation → **Judgment** → **Assurance**
→ **Authority** → **Execution** → **Evidence**

The proposed boundary

An agent's proposal describes a possible action; a model's judgment is evidence about it. Neither is, by itself, authority to mutate protected state. ACP is designed to prevent failures in generation or judgment from automatically carrying execution authority into protected state.

For readers	What this document provides
Architecture	The Evidence-to-Authority boundary, six-stage model, OpenKedge research stack, and one-action deployment path.
Protocol	A proposal-to-outcome lifecycle, message fields, grant schema, and exceptional transitions.
Security contract	Conditional invariants, trusted components, failure containment, and limits of stale-state protection.
Research plan	A minimal prototype, comparisons with strong existing authorization systems, and experiments that could falsify the claims.

What is established here

This is a **design specification, version 0.3**. It does not claim a deployed ACP, measured overhead, or improved safety in production. The strongest proposed safety property depends on complete gateway mediation and an atomic check of commit-relevant state; adapters lacking atomic enforcement must declare their reduced guarantee level. Reference monitors, policy decision/enforcement systems, and capabilities are established predecessors. ACP specifies how these classical abstractions compose with action-specific, heterogeneous, fresh, and potentially correlated evidence before an agent-proposed mutation receives execution authority. A conventional PDP/PEP that realizes the complete contract would be functionally equivalent.

Contents

1 The missing boundary

Suppose an infrastructure agent receives: "Reduce database spend without impacting production availability." It proposes removing a replica that appears idle. A semantic evaluator estimates that the action matches the request with probability 0.97. The probability estimate provides a helpful signal: it indicates how one model evaluated intent alignment within its prompt context. It does not, however, establish that the replica is currently idle, that failover capacity is sufficient, that the requesting principal holds requisite permissions on the target database, or that concurrent operations have not rendered the mutation unsafe.

The structural vulnerability is the implicit promotion from *judgment* (evidence) to *authority* (executable permission). An orchestration graph typically branches on a model score, and a tool adapter executes using broad, ambient credentials. Even an accurate model can operate on stale, unverified, or adversarial context. ACP makes this transition explicit, inspectable, policy-governed, and independently enforceable.

Definition 1 (Evidence-to-Authority boundary)**.** The Evidence-to-Authority boundary governs when and how evidence supporting an action is converted into narrowly scoped, verifiable authority permitting that *exact* action to mutate protected state. It cryptographically binds a canonical proposal, action-specific assurance obligations, the verified witnesses that discharged them, an active policy version, and commit-time state conditions to an independently redeemable execution grant.

Generation → **Judgment** → **Assurance** → **Authority**
→ **Execution** → **Evidence**

Each stage has distinct security semantics and operational boundaries. **Generation** proposes an intent, plan, tool call, delegated subtask, or external mutation. **Judgment** evaluates proposal semantics, scores intent alignment, or tests specific claims. **Assurance** determines whether the proposed action satisfies all policy-defined obligations using valid, fresh, and sufficiently independent witnesses. **Authority** mints an integrity-protected execution grant upon successful admission. **Execution** validates and redeems the grant at the protected resource boundary, mediating physical mutation. **Evidence** records the authoritative execution trace, gateway redemption receipt, and observed post-state outcome.

This architectural separation is meaningful only if it alters system behavior under adversarial or accidental failure:

- A compromised generator can construct malicious or malformed proposals, but cannot execute them directly.
- A faulty or compromised evaluator can return erroneous evidence, but cannot mint execution grants.
- An intercepted bearer grant remains strictly bounded to its canonical action, target resource, parameter predicate, validity interval, and single-use nonce.
- A stale grant fails redemption when commit-relevant state has changed, provided the underlying resource supports atomic precondition checking.

These failure-containment properties define the formal and empirical claims ACP must substantiate.

1.1 Why this boundary matters

The difference is the placement of authority, not the number of models in the loop:

Pattern	Path to a protected effect
Direct agent execution	Agent → Tool → long-lived credential → Effect
Judge-gated execution	Agent → model judge → Tool → Effect
ACP	Proposal → Evidence → Assurance → Grant → Gateway → Effect → trusted Outcome Evidence

While an LLM judge gate can filter ill-formed proposals, directly invoking privileged tools upon a positive score leaves the model's judgment in the de facto authorization role. ACP requires that evidence first satisfy policy-defined assurance obligations to achieve admission, after which an independent issuer mints a cryptographically bound grant for gateway redemption. The governing question is not whether an evaluator generally approves the plan, but whether verified evidence discharges all required obligations for that *exact* mutation under the active policy and current state.

Four separations

Solving is not verifying. Deliberation is not control. Judging is not authorizing. Authorizing is not executing.

Iterative reasoning, reflection, or multi-turn self-critique may improve proposal quality, but none establish an independent control boundary when the generator and evaluator share the same execution context, prompt lineage, or underlying weights.

1.2 What ACP adds beyond existing controls

Existing controls remain valuable and can implement parts of ACP. The table identifies the question each addresses and what a consequential stochastic action still requires.

Approach	What it establishes	Question still to resolve
Long-lived agent credentials	Workload identity and a reusable permission envelope	Does this specific proposed mutation have current, sufficient evidence and acceptable consequences?
RBAC / ABAC [1]	Static role- or attribute-based constraints over subject, resource, operation, and context	Which fresh semantic, deterministic, and operational witnesses satisfy this action's assurance obligations right now?
Conventional PDP / PEP [2, 3]	Architectural decoupling of policy decision from policy enforcement	A complete ACP contract can be implemented within this paradigm; absent ACP semantics, evidence freshness, correlation limits, cryptographic grant binding, and trusted outcome linkage remain ad hoc engineering choices.
LLM judge gate	Additional semantic assessment before execution	Does the model's output remain non-authoritative evidence, or does a positive score directly trigger privileged tool invocation?
Sandbox	Confinement of the agent runtime environment and observable side effects	Which external state mutations are authorized at the protected resource boundary?

Approach	What it establishes	Question still to resolve
Human approval	Human-in-the-loop review for high-consequence operations	Is the human approval cryptographically bound to the exact canonical action, parameters, verified state preconditions, and resulting execution?
ACP	End-to-end Evidence-to-Authority protocol: canonical proposal, obligation discharge, action-bound grant, gateway redemption, and outcome linkage	Assumes complete mediation, sound policy, and resource-side enforcement; it does not replace underlying authentication, isolation, or atomic resource guarantees.

The distinguishing design question is: given declared intent, a proposed action, delegated authority, current state, available semantic and deterministic evidence, known dependencies, and consequences of error, *what evidence is sufficient to authorize this exact mutation now?* ACP specifies how the answer becomes narrow, enforceable authority. It does not claim to invent authorization, capabilities, reference monitors, control planes, or zero trust.

1.3 Why a control-plane analogy helps

Software-defined networking separated control decisions from packet forwarding; OpenFlow provides one concrete protocol for programming switches [4]. ACP borrows the architectural discipline, not the packet model. Here the unit is a typed, state-changing operation, and the control decision must account for semantic intent, changing evidence, and noncommuting side effects. An action-bound grant plays a role analogous to a bounded rule enforced by a data-plane component. The analogy does not establish novelty or safety by itself.

Design objective

Prevent a proposal or semantic judgment from directly carrying the authority needed to mutate a protected resource.

2 Architecture at a glance

Figure 1 illustrates the logical control flow. While these functional stages can execute within a single process or across distributed services, their cryptographic identities, operational responsibilities, and credential boundaries remain strictly partitioned. Logically centralized control policies can be replicated to local decision nodes; ACP does not introduce a centralized runtime bottleneck on every request.

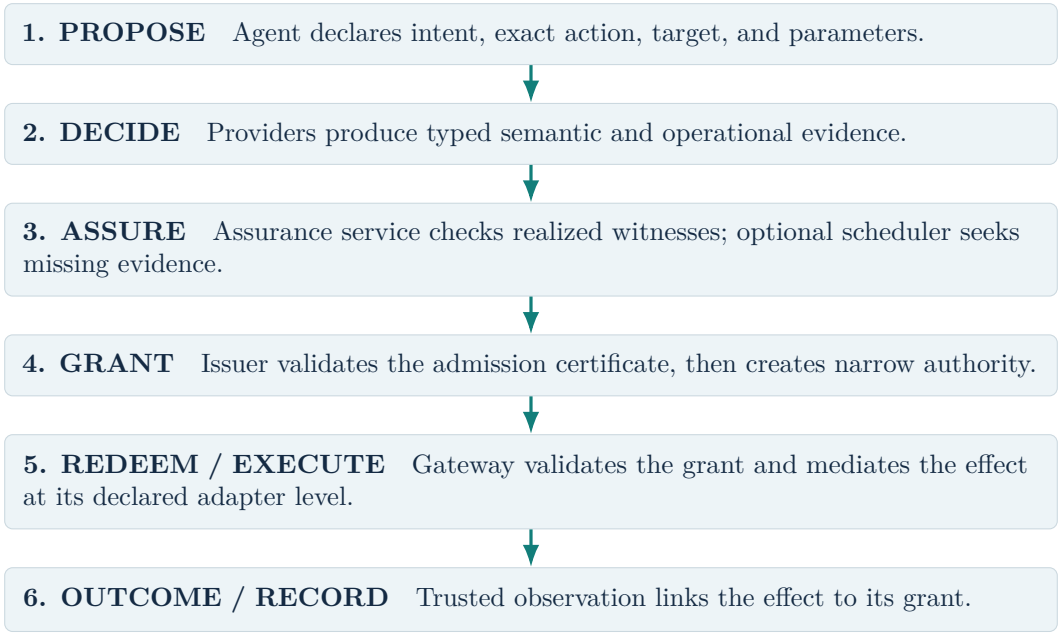

Figure 1: Logical path: providers supply evidence, assurance issues an admission certificate, and the grant carries issued authority. The gateway mediates each in-scope mutation with its backing resource credential.

2.1 Who owns which decision?

Component	Can produce	Cannot do by itself
Agent / generator	Canonical proposal, plan, parameter payload	Mint execution authority or directly access protected resources
Decision provider	Structured, typed evaluation evidence	Grant execution authority or invoke protected tools
Assurance service	Obligation resolution, witness verification, admission certificate (C_q)	Mint execution grants (G) or execute mutations
Grant issuer	Signed, action-bound execution grant (G)	Directly execute mutations or attest to execution outcomes
Execution gateway	Grant redemption receipt, mediated mutation call	Accept unadmitted proposals or direct agent prompts as authority
Outcome service	Cryptographically linked audit trail, observed outcome receipt	Issue execution grants or alter historical execution records

3 OpenKedge architecture synthesis

ACP composes several OpenKedge research mechanisms along the path from intent to protected effect (Figure 2). In this architecture: CAC and AAS define formal models for obligation specification and evidence acquisition; admission, issuance, and redemption define protocol roles; while the assurance service, optional scheduler, grant issuer, and execution gateway represent concrete runtime components.

OpenKedge implementations serve as reference designs; independent systems satisfying the protocol contracts achieve equivalent properties.

3.1 Identity: Persistent Cognitive Identity

Question: Who is acting, and what identity or delegation lineage does this action inherit? PCI models cognitive continuity, agent provenance, and delegation chains across substrate boundaries [5]. ACP incorporates this delegation lineage while strictly requiring authenticated workload identities at proposal submission, grant issuance, and gateway redemption. Epistemic identity continuity provides provenance context, but cannot substitute for cryptographic workload authentication.

3.2 Admission: Cognitive Admission Control

Question: What must be known before this consequential action may proceed? CAC defines consequence-conditioned assurance obligations for typed actions given current system state [6]. The assurance service evaluates realized witnesses against these obligations and issues an admission certificate C_q; a prospective or hypothetical acquisition plan is never sufficient for admission.

3.3 Acquisition: Assurance-Aware Semantic Scheduling

Question: Which evidence should be acquired, refreshed, diversified, or escalated under cost and latency constraints? AAS formalizes optimal evidence acquisition strategies for discharging CAC obligations under latency, cost, and epistemic risk constraints [7]. An optional evidence scheduler can orchestrate cheap deterministic checks first, refresh aging telemetry, or escalate ambiguous semantic assertions to higher-assurance providers. While scheduling optimizes the acquisition path, the assurance service independently verifies the resulting receipts at admission.

3.4 Dependence: Honest Quorum and epistemic fault domains

Question: Are supposedly independent witnesses vulnerable to the same cognitive failure? The Honest Quorum model formalizes Epistemic Byzantine Fault Tolerance for agentic systems, establishing exposure maps and witness-diversity constraints [8]. Multiple concurring evaluators do not constitute independent confirmation if they share pre-training corpora, fine-tuning recipes, retrieval corpora, telemetry feeds, or administrative control. Accounting for hidden epistemic correlations remains both a research challenge and an operational imperative.

3.5 Outcome: agent-native telemetry

Question: What actually happened after authority was exercised? Agent-native telemetry provides signed state-delta receipts and verifiable post-execution observations [9]. In this specification, we designate this telemetry role as ATP-T to avoid ambiguity with Agentic Transaction Processing (ATP). ACP strictly requires authoritative observations from the execution gateway or target resource; self-reported agent assertions of successful execution are non-authoritative.

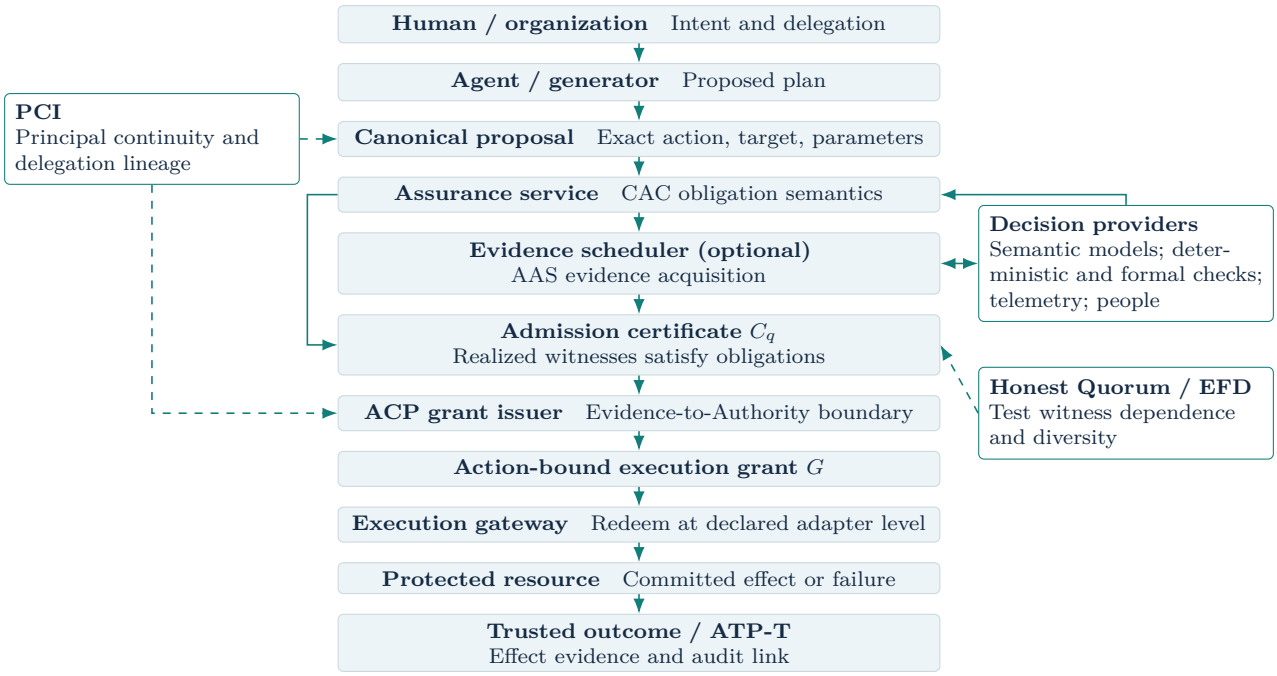

Figure 2: Logical roles, not required product names. The direct assurance-to-admission path bypasses the optional scheduler; providers can supply evidence directly. Dashed arrows show optional acquisition, identity, or witness context. C_q attests admission; G carries issued authority. The gateway holds the backing mutation credential; ATP-T is one option for outcome evidence.

3.6 Integration: the Agentic Control Plane

Question: How does satisfied assurance become enforceable authority over protected state? ACP establishes the formal binding between satisfied assurance and enforceable execution authority. It maps the canonical proposal, admission certificate (C_q), active policy version, commit-time state predicate, and authenticated principal into an action-bound execution grant (G). The execution gateway redeems this grant immediately prior to mutating protected state, validating preconditions and linking verifiable outcome receipts back to the authorization chain.

4 Where the Agentic Control Plane Applies

The following are illustrative applications of the architecture, not claims of deployed OpenKedge integrations. They show how the same boundary can support different evidence and authority profiles.

4.1 Infrastructure automation

An infrastructure agent proposes reducing the replica count for a production database cluster. Relevant evidence includes the principal's delegation lineage, target service-level objectives (SLOs), real-time traffic volume, failover headroom, failure-domain topology, concurrent deployment locks, and the current resource version. CAC policy specifies mandatory capacity and availability invariants; an AAS scheduler can refresh telemetry immediately prior to admission. The resulting grant authorizes exclusively the admitted replica-count mutation on the specified resource identifier. A Level A gateway adapter rejects

execution if the commit-time resource version differs from the admitted state; Level B or C adapters must explicitly declare their residual race window.

4.2 Security automation

A security agent proposes rotating a production service credential following an anomaly detection event. In this domain, evaluation delay exacerbates exposure risk, whereas erroneous revocation risks immediate service disruption. The assurance profile adapts accordingly: verifying credential identity and affected dependencies, assessing consumer readiness and blast radius, and activating emergency escalation pathways when policy permits. Authority is restricted to rotating the designated credential within a narrow validity window and for a single redemption. The authoritative outcome record must disambiguate successful rotation, failed rotation, and partial dependency disruption.

4.3 Financial or business workflow

A customer-support agent proposes issuing a refund or account credit. Required evidence spans transaction settlement status, customer and agent authorization tiers, real-time fraud risk scores, disbursement caps, dual-control (four-eyes) invariants, and mandatory human approval thresholds. The minted grant binds the precise account identifier, original transaction reference, credit amount, currency, and client-generated idempotency key. The payment gateway must report verified settlement effects, explicitly persisting an *unknown pending reconciliation* state whenever external clearinghouses have not confirmed finality.

These examples share a common protocol, not a uniform policy. The required witnesses, delegation rules, temporal validity, and compensation or rollback expectations scale with consequence, uncertainty, and reversibility.

5 Start with One Consequential Action

ACP is proposed as an **enforcement and control layer**, not another agent framework. An organization can select one protected mutation and route it through an ACP-compatible boundary:

Existing agent framework → ACP boundary
→ existing API, cloud, Kubernetes, database, or enterprise service

Candidate first actions include production capacity scaling, deployment rollbacks, IAM policy edits, credential rotation, resource teardown, and direct database mutation. The existing agent framework above the boundary does not need replacement. However, its tool-invocation architecture and credential paths must adapt: the agent submits a canonical proposal, while a mediation gateway holds the backing mutation credentials and enforces the issued grant. Any ambient or direct access credentials to the target resource must be revoked or explicitly excluded from the safety claim.

A minimal initial engagement selects a single mutation; establishes complete mediation and isolated credential custody; defines the canonical action schema and assurance obligations; implements realized-witness admission and independently verified narrow grants; enforces them through a dedicated gateway; declares the adapter guarantee level (Section 9); and systematically validates the boundary using the direct-bypass and concurrency test suites (Section 10). A Level B or C adapter serves as a viable

adoption baseline only with its narrower state guarantees made explicit. This represents an incremental adoption path, not a claim of existing commercial deployment.

6 System model and trust boundary

ACP governs a declared universe of state-changing operations across protected systems. A deployment maintains authoritative registries of requesting principals $\mathcal{P}$, canonical typed actions $\mathcal{A}$, protected resources $\mathcal{R}$, and versioned policies Π_v. Every protected action possesses a strict canonical schema: semantic aliases, whitespace variations, or reordered fields cannot alter how the assurance service or gateway interprets the action. The resource registry maps each canonical target resource to its designated mediation gateway and underlying backing API.

6.1 A proposal is an object, not a credential

A proposal represents a structured, non-authoritative intention to act:

$$q = \langle id, p, i, a, r, \theta, d, x, h_{s_0} \rangle. \tag{1}$$

Here id is a globally unique proposal identifier, $p \in \mathcal{P}$ is the authenticated principal, i declares high-level intent, $a \in \mathcal{A}$ is the canonical typed action, $r \subseteq \mathcal{R}$ designates the target resource set, θ denotes the canonical parameter payload, d specifies causal dependency references, x provides operational execution context, and h_{s_0} is a cryptographic digest or version identifier of baseline observed state s_0. A proposal can be admitted, denied, remediated, or reissued, but conveys zero authority to execute against protected state.

The assurance service evaluates proposal q against active policy Π_v and an acquired evidence set E. We denote by $\text{Admitted}_{\Pi_v}(q, E, s_a)$ its evaluation verdict at admission-time state s_a. This decision is strictly decoupled from $\text{Committed}(q, s, s')$, which denotes that the target resource transitioned from pre-state s to post-state s'. A provider evaluation constitutes a candidate evidence item; a *witness* is an evidence item verified and accepted to discharge a specific assurance obligation. Admission can occur without subsequent grant issuance or execution; an attempted execution may abort or fail; and post-execution outcomes may remain temporarily uncertain pending reconciliation. The control plane preserves these state distinctions as disjoint audit receipts.

6.2 What can go wrong?

ACP considers four propagation layers:

1. **Compromised intelligence:** prompt injection, malicious retrieved context, poisoned memory, a compromised generator, or an over-broad plan.
2. **Incorrect judgment:** uncaught hallucinated parameters, stale telemetry, erroneous semantic evaluation, model scoring errors, or correlated misjudgments among structurally dependent providers.
3. **Invalid authority:** policy drift, forged or stolen grants, confused-deputy exploitation, replay attacks, privilege expansion during delegation, or grant issuance without satisfied obligations.

4. **Enforcement bypass:** direct credential leakage to agents, unauthorized tool paths, unmediated gateway bypass, missing precondition checks in adapters, or race conditions between state validation and non-atomic execution.

The architecture is designed to prevent failures in generation or judgment from automatically translating into execution authority over protected state. It cannot, however, prevent compromise of the issuer's private signing keys or compensate for an unmediated execution path. Nor can formal control transform an unsound policy, missing dependency model, or falsified ground-truth evidence into safe execution.

6.3 Trusted base and coverage

The safety guarantees hold under explicitly stated trust assumptions: authenticated principals and providers; integrity-protected policy distribution; protected private keys for the assurance service and grant issuer; deterministic request canonicalization and resource mapping; exclusive custody of mutation credentials by the gateway; durable replay state; and a gateway that mediates *every* in-scope mutation. For every canonical (r, a) pair, a versioned *coverage inventory* documents the resource identifier, covered actions, designated gateway, backing API, mutation-credential owner, known alternate credential routes, unmanaged mutation vectors, enforcement mechanism, and declared adapter guarantee level. Coverage must be empirically verified via resource-side access control audits and direct penetration tests, rather than inferred from high-level orchestration diagrams. Administrative accounts, CI/CD pipelines, or auxiliary agents capable of mutating the same resource surface must either pass through the mediation gateway or be explicitly excluded from safety claims. **ACP safety claims apply exclusively to the mutation surface for which complete mediation is empirically verified.** While read-only observations may utilize unmediated paths, any evidence acquisition operation entailing side effects constitutes a protected action requiring its own assurance and mediation.

To guarantee state-valid execution at commit time, the gateway requires an atomic resource-side synchronization primitive: compare-and-swap (CAS), conditional PUT, transactional commit, or cryptographic fencing tokens. A pre-state digest verified during admission provides necessary freshness evidence, but cannot prevent Time-of-Check to Time-of-Use (TOCTOU) races against backing APIs lacking atomic precondition enforcement. For non-atomic APIs, ACP provides pre-execution revalidation and post-hoc reconciliation, but the adapter must explicitly declare a degraded guarantee level.

6.4 Credential custody and compromise

The requesting agent, decision providers, and assurance service hold no credentials capable of mutating protected resources. The grant issuer possesses signing keys to issue execution grants, but has no access to backing resource credentials. The execution gateway alone holds and exercises credentials for backing APIs; target resources authenticate the gateway or an equivalent reference monitor. Any configuration granting direct mutation credentials to an agent or evaluator violates complete mediation and immediately invalidates authorization guarantees for that surface.

Component compromise propagates strictly downstream: a compromised agent can submit adversarial proposals; a compromised provider can generate deceptive evidence; a compromised assurance service can falsely attest admission; a compromised issuer can forge execution grants; and a compromised gateway can directly execute unauthorized mutations. Consequently, the assurance service, grant issuer, execution gateway, policy distribution pipeline, replay storage, and resource-side access controls constitute the Trusted Computing Base (TCB) for their respective claims. The Evidence-to-Authority architecture

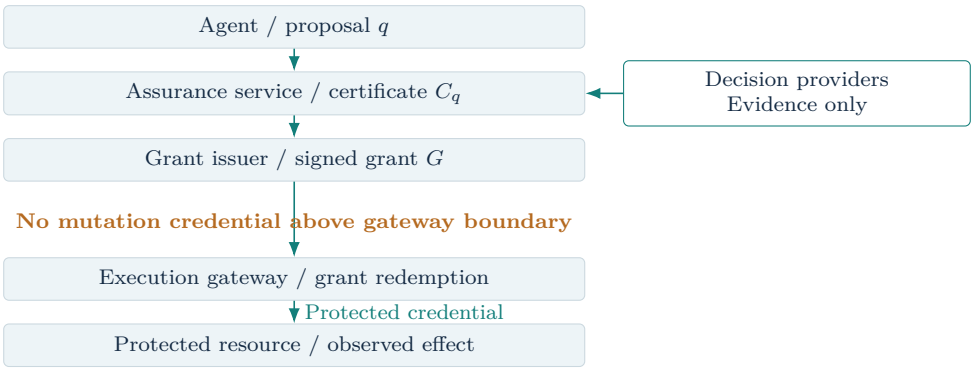

Figure 3: Trust boundary and credential custody. Provider arrows carry evidence, C_q attests admission, and G carries narrow authority. Only the gateway can exercise the backing mutation credential.

limits the lateral and downstream propagation of upstream failures; it does not render the system immune to subversion of TCB components themselves.

7 Control-plane protocol

Every control-plane message carries a protocol version, unique message identifier, authenticated sender identity, timestamp, canonical payload digest, and cryptographic integrity envelope; proposal identifiers and parent lineage references are attached where applicable. These message types specify logical protocol transitions rather than mandatory network boundaries. The standard execution lifecycle proceeds through nine phases:

$$\begin{aligned}\mathsf{REGISTER} \to \mathsf{POLICY\_SYNC} \to \mathsf{PROPOSE} \to \mathsf{DECIDE} \\ \to \mathsf{ASSURE} \to \mathsf{GRANT} \to \mathsf{REDEEM} \\ \to \mathsf{OUTCOME} \to \mathsf{RECORD}\end{aligned}$$

Every lifecycle transition generates an immutable, cryptographically verifiable receipt. Any modification to a proposal's action, target resource, parameter set, or bound state requires re-initiating admission; execution grants are immutable and cannot be modified post-issuance.

The decisive authorization chain is:

$$\begin{aligned}\langle q, E, \Pi_v \rangle \longrightarrow C_q \longrightarrow G \longrightarrow \mathrm{Redeem}(G) \\ \longrightarrow \mathrm{Effect} \longrightarrow \mathrm{Outcome}\,.\end{aligned} \tag{2}$$

Admission verifies that all required assurance obligations are discharged by valid witnesses ($\langle q, E, \Pi_v \rangle \longrightarrow C_q$); the independent verification step $C_q \longrightarrow G$ mints executable authority; and redemption $\mathrm{Redeem}(G)$ enforces that authority at the protected resource boundary.

7.1 Message contract

Message	Required content	Receiver check and effect
REGISTER	Principal, provider, gateway, or resource ID; canonical action schema; verification key; protocol version	Authenticate registrar; install versioned identity record and verified schema.
POLICY_SYNC	Signed Π_v, activation timestamp, supersession rule, revocation epoch, compatible schema and gateway versions	Check signature, monotonic version and epoch, and activation; acknowledge installation or fail closed.
PROPOSE	Canonical q from Eq. 1; authenticated principal signature; dependencies and baseline state references	Resolve exact action and target; freeze proposal digest $H(q)$. Yields zero execution authority.
DECIDE	Provider ID and version, question schema Q_D, state view digest, typed result E_D^*, timestamp, provenance, exposure domains	Validate schema, signatures, and provenance; append candidate evidence item. Yields zero authority.
ASSURE	Proposal digest $H(q)$, active Π_v, obligation set Ω, realized witness manifest E^*, state binding b_s, discharge results	Assurance service evaluates discharge predicates; issues C_q if and only if all obligations are SATISFIED.
GRANT	Admission certificate C_q, requested scope, caller identity, target gateway audience *aud*	Issuer validates certificate integrity, service identity, proposal binding, policy currency, freshness, and scope; mints signed G.
REDEEM	Execution grant G, exact canonical request payload, caller identity, gateway and resource IDs	Gateway validates issuer signature, audience, caller, scope, policy compatibility, revocation epoch, nonce, and state predicate φ_s.
OUTCOME	Attempt ID, observed state delta, error code, or explicit *unknown* status; authoritative resource version if observed; observation timestamp	Distinguish committed, aborted, and unreconciled effects.
RECORD	Cryptographically chained digests of proposal, evidence receipts, admission certificate, grant, redemption, and outcome	Store a queryable audit record with explicit gaps and retention metadata.

7.2 Exceptional transitions

DENY conveys unsatisfied obligation identifiers, structured reason codes, active policy version, and the evaluation trace. **REMEDIATE** specifies unmet obligations alongside a constrained remedy schema and parent proposal reference; any consequential remediation action must undergo its own admission lifecycle. **ESCALATE** re-routes unresolved or ambiguous obligations to a higher-assurance provider class, multi-provider committee, or authorized human reviewer. **REFRESH** re-evaluates aging evidence against an updated state digest while preserving historical receipts. **DELEGATE** requests a child grant from the issuer, enforcing strict monotonic scope attenuation relative to the parent grant. **REVOKE** invalidates a grant identifier, assurance signing key, or issuer key, specifying an activation timestamp, reason code, and monotonically increasing revocation epoch; gateways enforce revocation synchronously or within a declared bounded-staleness window Δ_{rev}.

System outages cannot be treated as affirmative verdicts. If a decision provider is unavailable, affected obligations evaluate to UNKNOWN. Under default-deny semantics, admission is withheld unless an alternate qualified provider supplies valid evidence, an authorized human reviewer attests to the condition, or the active policy explicitly defines a safe fallback rule. If the assurance service or grant issuer is unreachable, no new execution grants can be minted. A gateway may redeem an existing valid grant only if its synchronized policy and revocation state are within their declared freshness bounds Δ_{sync}; otherwise,

it must fail closed. If a mutation commits but the outcome observation is delayed or lost, the system persists an *unknown pending reconciliation* state, strictly rejecting client- or agent-reported assertions of success. Following a crash, a gateway must recover durable nonce state before accepting retries; operations with uncertain external effects require idempotent retry tokens or manual reconciliation before re-issuance. ACP does not claim exactly-once semantics without backing resource support.

7.3 Policy activation, supersession, and revocation

Each signed policy version specifies its activation time, issuance eligibility, compatibility rule for outstanding grants, revocation epoch, and maximum synchronization age. The issuer and gateway durably retain a monotonic high-water mark and reject rollback to an older signed policy or epoch. The issuer checks that the certificate's policy is issuance-eligible at issuance time; otherwise it requires re-admission. At redemption the gateway requires

$$\mathrm{Compat}(\Pi_{v_{\mathrm{issued}}}, \Pi_{v_{\mathrm{current}}}, G) = \mathrm{true}, \tag{3}$$

in addition to grant validity. Here "current" is the latest policy the gateway has authenticated within its permitted synchronization age; an update unknown during the propagation window cannot be enforced instantaneously. An unspecified compatibility rule fails closed. A versioned rule may retain an older grant until expiry, invalidate it after activation and the declared propagation bound, or permit it only for compatible actions; the choice is explicit policy, not an implicit assumption.

For example, after admission and grant issuance under Π_{18}, activation of Π_{19} does not by itself decide the grant's fate. Its signed supersession rule determines whether $\mathrm{Compat}(\Pi_{18}, \Pi_{19}, G)$ holds. **POLICY_SYNC** distributes the new policy and epoch; **REVOKE** names grants or keys invalidated by an emergency update. Emergency invalidation has a declared propagation bound Δ_{rev}, not instantaneous global effect. A gateway that cannot confirm a sufficiently fresh epoch by its deadline stops redeeming affected grants. An already committed effect cannot be revoked. Key rotation and policy supersession use the same monotonic, signed distribution discipline.

7.4 Minimum ACP Conformance

This is an architecture-level checklist; wire interoperability still requires pinned schemas and cryptographic profiles. To claim ACP compatibility for a declared mutation surface, an implementation **MUST** provide:

1. Canonical typed proposals and authenticated principal/resource identity.
2. Separation of decision evidence from executable authority, with versioned assurance policy.
3. Admission over realized witnesses and an integrity-protected certificate bound to proposal, policy, obligations, evidence, and state.
4. Independent issuer validation of that certificate and an action-bound, integrity-protected grant.
5. Gateway enforcement of exact action, resource, parameters, audience, caller binding where required, policy compatibility, and revocation.
6. Durable replay/redemption control, declared state-validation level, and trusted outcome recording with explicit unknown states.

7. A coverage inventory and complete mediation for every resource/action pair included in its safety claim.

These grouped requirements are mandatory within the claimed surface. AAS scheduling, PCI, Jev, multiple semantic models, human review, ATP-T-specific telemetry, formal verification, and epistemic-diversity rules are optional mechanisms or policy choices. An independent system can implement the mandatory contracts without using OpenKedge components.

8 From evidence to authority

8.1 Judgment is typed evidence

A decision provider answers a versioned question Q_D evaluated over an authenticated state view S_D:

$$\text{evaluate}(S_D, Q_D) \longrightarrow E_D^*. \tag{4}$$

Here D identifies the provider, S_D denotes its observed state view, Q_D is the versioned question schema, and E_D^* represents the resulting candidate evidence item. Each evidence item identifies its provider and version, question schema, state digest, typed result, observation timestamp, provenance trace, and known shared-failure domains. Probability distributions and calibration metadata are optional annotations. Deterministic invariant checks, human attestations, model scores, and formal proofs can share the same transport envelope without conflating their underlying semantics. The active policy defines which evidence types and polarities are admissible to discharge each obligation.

For the database replica example, a semantic provider evaluates whether "reduce spend without impacting availability" permits removing a specific replica. A deterministic provider verifies minimum replica thresholds and failure-domain spread. Telemetry providers report recent load and failover health. A human operator may attest to an authorized maintenance window. ACP adapters can integrate deterministic verifiers, Jev, independent LLM evaluators, formal theorem provers, telemetry sinks, or human review queues. Jev serves as one candidate semantic provider if its version and question schemas are pinned; the architecture neither mandates it nor makes vendor-specific performance claims. **Providers produce non-authoritative evidence; an affirmative evaluation never conveys mutation authority.**

8.2 Assurance is action-specific

Let $\Omega_{\Pi_v}(q, s_a)$ be the obligation set generated by policy Π_v for proposal q given admission state s_a. Admission requires every obligation to be discharged by valid, realized witnesses:

$$\text{Admitted}_{\Pi_v}(q, E, s_a) \equiv \bigwedge_{\omega \in \Omega_{\Pi_v}(q, s_a)} \text{Discharge}(\omega, E, q, s_a). \tag{5}$$

Discharge evaluates witness authenticity, question and state bindings, evidence class, polarity, freshness limits, and epistemic diversity constraints. It returns SATISFIED, VIOLATED, or UNKNOWN; an admission certificate can be minted if and only if every obligation evaluates to SATISFIED. CAC formalizes these obligation semantics [6]; the assurance service executes their evaluation. AAS defines optimal acquisition and escalation policies under cost and latency bounds [7]; an optional scheduler orchestrates that strategy. A prospective scheduling plan does not constitute obligation discharge; the assurance service evaluates realized evidence receipts at admission time.

Predicate correspondence. Evidence must substantiate the exact predicate mandated by an assurance obligation, not an adjacent or relaxed assertion. An affirmative result for predicate P_e cannot discharge an obligation requiring P_r unless policy explicitly proves or defines $P_e \Rightarrow P_r$. Semantic similarity or provider confidence scores cannot substitute for formal predicate correspondence.

Definition 2 (Admission certificate)**.** An admission certificate is an integrity-protected attestation minted by an authorized assurance service:

$$C_q = \langle cid, H(q), v, H(\Omega_{\Pi_v}(q, s_a)), H(E^*), b_s, F, \\ t_a, t_e, aid, aver, k_A, \sigma_A \rangle. \tag{6}$$

Here cid is a unique certificate identifier; $H(q)$ is the canonical proposal digest; v is the active policy version; s_a is the system state evaluated at admission; $\Omega_{\Pi_v}(q, s_a)$ is the resolved obligation set; E^* is the manifest of realized witnesses discharging those obligations; b_s cryptographically binds the evaluated state version and expected-state predicates; F records applicable per-witness freshness bounds; t_a, t_e define admission and certificate-expiration timestamps; $aid, aver$ identify the assurance service and its implementation version; and k_A, σ_A denote the service's signing key identifier and digital signature over all preceding fields. Possession of C_q attests that the assurance service verified that all obligations required by Π_v for q were satisfied under evidence E^* at time t_a.

C_q does not attest that the action remains safe indefinitely, that evidence remains fresh, that policy remains current, that state remains unchanged, or that execution occurred. The grant issuer must independently verify certificate integrity, authorized service identity, proposal and policy bindings, temporal validity, requested-scope containment, and active revocation state before minting G. The issuer may trust the authorized assurance service's discharge verdicts without re-executing semantic evaluations; that service remains in the Trusted Computing Base.

8.3 Consequence-conditioned assurance

The same evidence threshold is inappropriate for every operation. The following progression is a policy illustration, not a theorem or a universal default:

Action	Illustrative assurance profile
Read deployment status	Read-only access policy; no mutation grant.
Restart a development container	Authenticated principal and deterministic policy checks may suffice.
Reduce production capacity	Current telemetry, intent interpretation, availability invariant, and commit-time state check.
Delete a production database	Strong heterogeneous evidence, deterministic dependency and backup invariants, and possibly human approval.

Policy principle

Assurance should scale with consequence, uncertainty, and reversibility. CAC defines the obligations; an assurance service evaluates them. AAS defines acquisition strategy; an optional scheduler may implement it.

Multiple model votes do not automatically supply independent support. Two evaluators may share training data, a telemetry feed, a prompt template, or an operator. The witness manifest records

known exposure domains, and the policy can demand structurally diverse sources. Agent-controlled text, retrieval, or telemetry is not an independent witness merely because a provider repeats it; provenance and source authority are checked per obligation. A policy interpreted only by the agent being governed would permit self-approval and cannot discharge an independent assurance requirement. Unknown common dependencies remain a risk and are measured empirically rather than assumed away.

8.4 The grant is the authority object

Definition 3 (Action-bound execution grant)**.** An execution grant is an integrity-protected capability:

$$G = \langle gid, H(q), H(C_q), p, a, r, \Theta, H(E^*), v, [t_0, t_1], \\ \varphi_s, \delta, n, aud, k_I, \sigma_I \rangle. \tag{7}$$

It binds a unique grant identifier gid; proposal digest $H(q)$; admission certificate digest $H(C_q)$; requesting principal p; canonical action a; target resource set r; parameter constraint predicate Θ; witness manifest digest $H(E^*)$; governing policy version v; validity window $[t_0, t_1]$; commit-time state predicate φ_s; delegation constraints δ; single-use redemption nonce and count limit n; target gateway audience aud; issuer key identifier k_I; and issuer digital signature σ_I authenticating all preceding fields.

For comparison, "modify production infrastructure" is a broad credential, not an ACP grant. The following *illustrative* grant authorizes one admitted mutation; the values are an example, not an issued token:

Bound field	**Illustrative value**
Principal (p)	`deployment-agent-17`
Action and resource (a, r)	`SetReplicaCount` on `prod/orders/db-7`
Allowed parameters (Θ)	`replicas == 2`
Policy version (v)	`production-capacity-v18`
Expected state (φ_s)	`resourceVersion == 88193`
Validity window ($[t_0, t_1]$)	`[10:40:00, 10:42:05 UTC]`
Redemption (n)	One use ($n = 1$); bound gateway audience and nonce
Evidence / admission	Witness-manifest digest $H(E^*)$; certificate digest $H(C_q)$

The agent does not receive general authority to modify the database. It receives, or causes to be issued, authority for one admitted mutation under specified conditions. The example presumes that the resource can enforce the version check at commit; otherwise that field supports revalidation but cannot by itself prevent a race.

This is a narrowly scoped capability, not a new kind of cryptographic primitive [10]. ACP's proposed contribution is the evidence-conditioned issuance and redemption lifecycle. Generic cloud or database credentials remain at the gateway. A high-risk grant may also be bound to a proof-of-possession key, not just bearer possession. Serialization, canonicalization, hash and signature algorithms, key rotation, and gateway audience are versioned wire-contract details.

8.5 Redeeming the exact action

At redemption time, the execution gateway verifies the complete authorization contract:

1. **Issuer validity:** authenticates signature σ_I under key k_I and confirms the issuer is authorized for action a and resource r;

2. **Audience and caller binding:** ensures the gateway matches aud and that the invoking caller matches principal p (or authorized delegation lineage);
3. **Scope matching:** verifies that the runtime invocation matches canonical action a, target resource r, and parameter predicate Θ;
4. **Temporal validity:** confirms the current timestamp falls strictly within validity window $[t_0, t_1]$;
5. **Policy compatibility and revocation:** verifies $\text{Compat}(\Pi_{v_{\text{issued}}}, \Pi_{v_{\text{current}}}, G)$ and ensures the grant or signing key has not been revoked in the active revocation epoch;
6. **Replay protection:** checks and durably expends redemption nonce n, preventing duplicate submission;
7. **State validity:** evaluates commit-time predicate φ_s against current resource state at the adapter's declared guarantee level.

A grant is *issuer-valid* if authentic and unrevoked; *policy-valid* if compatible with the gateway's active policy epoch; and *state-valid* if predicate φ_s holds over commit-relevant state. None of these mechanical checks guarantees that the action is semantically optimal. Upon successful validation, the gateway durably commits the nonce reservation and executes the mutation; only a Level A adapter executes the state check and mutation atomically. The gateway logs an attempt receipt, mediates execution, and captures the authoritative outcome. If a grant is issued but expires unredeemed, the audit log records its expiration rather than presuming execution.

For delegated execution, any child grant must remain strictly monotonic, attenuating authority across actions, resources, parameter predicates, audience, validity intervals, and redemption counts. Revocation halts subsequent redemptions across all gateways within declared propagation bound Δ_{rev}. Revocation cannot undo an effect already committed to protected state. While offline gateway validation maximizes availability during control-plane partitions, it widens the vulnerability window to revoked authority; this operational trade-off must be explicitly governed by deployment policy.

9 Safety properties and failure containment

ACP specifies security and safety properties over execution traces of protected effects. These properties constitute conditional engineering contracts conditioned on explicitly stated trust and mediation assumptions, not claims that models or policies invariably select optimal actions.

Authorization safety requires that every committed, in-scope mutation was authorized along the stipulated control path and executed under an issuer-valid, policy-valid grant matching the exact canonical action, target resource, and parameter predicate, under complete mediation. **Semantic correctness (decision quality)** evaluates whether that authorized mutation faithfully reflects true human intent and safely accommodates real-world domain dynamics, including unmodeled environmental dependencies. ACP formally targets authorization safety; it does not and cannot guarantee semantic infallibility. Corrupted evidence, flawed policy logic, or incomplete dependency specifications can result in the admission of an undesirable action. **ACP enforces rigorous control and auditability over the Evidence-to-Authority transition; it does not render underlying evidence sources or policy models infallible.**

Property	Required meaning
No direct execution	A proposal alone cannot trigger a protected mutation.
Judgment non-authority	Decision evidence alone cannot trigger a protected mutation.
Admission before grant	Every issued grant has a traceable all-satisfied admission certificate under an active policy version.
Grant-bound execution	Every committed protected action has a successfully redeemed grant for that exact canonical action.
Evidence-bound authority	A verifier can reconstruct which evidence and obligations justified issuance.
Scope monotonicity	A delegated child grant cannot expand its parent's authority across any dimension.
Revocability	A revocation effective before redemption blocks that redemption within declared propagation bound Δ_{rev}.

The core authorization invariant can be expressed as:

$$\text{Committed}(q, s, s') \implies \exists G : \text{Redeemed}(G, q) \wedge \text{Valid}_{\text{state}}(G, s). \tag{8}$$

Proposition 1 (Conditional grant-bound and state-valid execution)**.** Assume a declared Level A surface where:

1. **Complete mediation:** every state-changing operation on resource $r \in \mathcal{R}$ passes through a designated, unbypassable execution gateway;
2. **Strict gateway enforcement:** the gateway rejects every request lacking an authentic, issuer-valid, and policy-compatible grant G binding the exact canonical request (p, a, r, θ);
3. **Durable replay protection:** redemption nonce state is persisted before mutation dispatch, precluding duplicate execution; and
4. **Atomic state validation:** the resource or gateway atomically evaluates the grant's commit-time state predicate φ_s concurrently with mutation execution.

Then every committed protected mutation $(s \to s')$ corresponds to a validly redeemed grant whose authorization scope covers the exact action, parameters, and commit-time state s.

Proof sketch. Under complete mediation, no execution path reaches protected resource r bypassing the gateway. The gateway admits dispatch only if grant G is issuer-valid, policy-valid, and its nonce unexpended. The durable nonce reservation guarantees at-most-once redemption per grant. Finally, Level A atomicity guarantees that condition $\varphi_s(s)$ holds at the precise instant of state mutation. Hence, no ungranted, replayed, or state-invalid mutation can commit on a Level A surface.

Non-claims and adapter boundaries. The proposition does not establish exactly-once external effects under network partitions or unconfirmed retries, nor does it guarantee the absence of semantic errors arising from faulty policies or misleading provider evidence. For Level B (revalidated) and Level C (reconciliation) adapters, Eq. 8 guarantees grant-boundedness, but does *not* guarantee commit-time state validity against concurrent races.

9.1 Adapter guarantee levels

Each ACP adapter declares the guarantee it can enforce for each protected action. These levels describe state-bound execution, not the quality of admission evidence:

Declared level	Enforcement and remaining limit
A: Atomic state-bound	The resource or gateway atomically evaluates the grant's commit-relevant state predicate φ_s with the mutation; stale-state commits against that predicate are rejected under stated assumptions.
B: Revalidated	The gateway rechecks state immediately before the API call, but the check and mutation are not atomic; a race can still commit.
C: Detection and reconciliation	The backing API cannot reliably enforce the precondition. The gateway validates grant scope and records the attempt, then observes or reconciles the effect; a state-violating commit can occur.

All levels require grant validation, durable replay control, outcome status, and complete mediation for the claimed surface. If an API cannot constrain the action or resource to the grant's exact scope, that action is outside ACP conformance until an enforcement mechanism exists. A Level C integration cannot claim commit-time prevention merely because it logs a mismatch afterward.

9.2 Where the guarantee stops

The proposition fails if an agent has a second credential path, a resource alias bypasses canonical matching, nonce state is lost on restart, or an adapter claims Level A while checking a condition before a non-atomic external call. A pre-state digest cannot solve a TOCTOU race by itself. Levels B and C make weaker state claims; they do not weaken the requirement to validate grant scope. Each coverage entry identifies its actual level.

The same distinction applies to audit. Signed and linked receipts can make a completed authorization trace reconstructible. If grant issuance or the effect commits while the record service is unavailable, a transactional outbox or reconciler must close the gap. Until then, the trace should report *unknown* and preserve the missing segment. Outcome evidence should come from the gateway or protected resource rather than the agent's own assertion of success.

What ACP does not solve

ACP does not make incorrect policies correct, incomplete dependency models complete, false evidence true, compromised trust roots trustworthy, bypassable gateways safe, or non-atomic APIs atomic. Within a completely mediated surface, ACP prevents failures in generation or judgment from automatically carrying execution authority into protected state. This authorization safety does not guarantee semantic correctness or decision quality.

Practical test

Try to mutate an in-scope resource without a matching grant, with a stolen or replayed grant, or after changing a commit-relevant version. A successful direct or replayed effect defeats the authorization claim; a stale-state effect defeats a Level A claim.

10 Prototype and evaluation plan

The initial prototype must implement a minimal end-to-end vertical slice rather than a general-purpose agent framework. It targets two concrete actions in an isolated test environment: resizing a database replica and adjusting a deployment replica count. The requesting agent constructs and transmits a canonical proposal. An assurance service implementing CAC semantics invokes both deterministic and semantic decision providers; an optional scheduler implementing AAS semantics orchestrates the acquisition and selective refresh of required evidence. Upon verifying that all obligations are discharged, the assurance service issues signed admission certificate C_q. An independent grant issuer validates C_q, verifies policy currency, and mints signed grant G. The execution gateway, maintaining exclusive custody of mutation credentials, validates G and redeems it against the target resource version. An append-only evidence store persists the authoritative outcome alongside the adapter's declared guarantee level.

The prototype requires versioned schemas for proposals, questions, candidate evidence, admission certificates, policies, grants, and outcome receipts; a resource dependency and coverage registry; key management and rotation for assurance, issuer, and gateway components; a durable, crash-resilient store for nonces and revocation epochs; and an independent trace verification engine. Jev can serve as a semantic provider adapter if exact, version-pinned access is available. Core security and protocol benchmarks must execute independently without Jev or any single proprietary vendor dependency.

10.1 Comparisons that matter

The evaluation should compare four configurations under identical workloads and policy specifications:

1. **Direct tools:** agent-generated tool calls execute directly using ambient credentials.
2. **Judge gate:** a semantic model inspects and approves or rejects tool calls immediately prior to execution.
3. **Hardened PDP/PEP baseline:** an industry-standard policy decision and enforcement architecture with real-time state querying (e.g., an Open Policy Agent gateway).
4. **Complete ACP:** full Evidence-to-Authority lifecycle, including canonical proposals, structured witness manifests, signed admission certificates, action-bound capability grants, gateway redemption, and cryptographically linked outcome receipts.

The third baseline is essential. Outperforming unconstrained agent execution or a naive single-prompt LLM judge provides negligible evidence of architectural contribution. ACP must demonstrate measurable containment advantages over a properly engineered enterprise authorization baseline under the same policy intent. Should a sufficiently extended PDP/PEP realize the entire Evidence-to-Authority specification, it is functionally equivalent; empirical evaluations must explicitly report such equivalence. Controlled ablation experiments should isolate the individual marginal utility of heterogeneous witness diversity, pre-execution evidence refresh, commit-time state binding, and single-use redemption nonces.

10.2 Measures and falsifiers

Failure events must be categorized into disjoint, rigorously defined failure classes:

- **Unauthorized commit:** a protected mutation executes without a valid, gateway-redeemed grant corresponding to the exact canonical action, resource, and parameters.

- **Contract-violating commit:** a mutation executes in breach of an explicit policy obligation, temporal validity window, or commit-time state predicate φ_s, regardless of whether a certificate or grant was minted erroneously.
- **Semantic failure:** a mutation fully satisfies the formal protocol and authorization path, yet produces undesirable operational outcomes under an external domain oracle due to specification flaws, erroneous evidence, or unmodeled dependencies.

Empirical evaluations must report counts and conditional rates for each failure mode under explicitly specified oracles, rather than aggregating results into an undefined "unsafe execution" metric.

Question	Measurement or deliberate attack
Control-plane overhead	Median, 95th, and 99th percentile latency; throughput under concurrency; CPU, memory, grant size, and signature verification overhead.
Commit failures by class	Under prompt injection, poisoned evidence, malicious retrieval, broad parameters, stale state, and compromised providers, count unauthorized, contract-violating, and semantic failures separately.
False denial	Legitimate operations blocked, by reason code and workload stratum.
Escalation efficiency	Fraction resolved by deterministic check, cheap semantic provider, multiple witnesses, stronger verifier, or human review; cost and latency by path.
Correlation	Paired failures for same-family judges, heterogeneous providers, and deterministic cross-checks under shared misleading inputs.
TOCTOU	Concurrent modification of commit-relevant state versions following grant issuance; measure race detection across Level A, B, and C adapters.
Replay and theft	Concurrent duplicate redemption, crash-recovery replay tests, cross-principal theft, wrong gateway, and parameter tampering.
Evidence completeness	Blind third-party verification of authorization lineage, obligation satisfaction, and outcome linkage from persistent receipts.
Availability	Network partitioning of providers, issuer, or revocation service; measure denial rates, backlog, and bounded-staleness window enforcement.

The primary security metric is Pr(unauthorized commit | injected fault); contract violations and semantic failures must be reported as distinct conditional rates stratified by fault category and adapter guarantee level. The primary usability metric is the false rejection rate Pr(deny | legitimate proposal). For epistemic dependency analysis, evaluations must estimate the joint error probability $\Pr(F_{\text{controller}} \mid F_{\text{generator}})$ across paired trials, rejecting consensus agreement as a proxy for ground-truth correctness. Published results must document exact confidence intervals, corpus generation methodology, policy specifications, model checkpoints, question schemas, raw receipts, and measurement timestamps. Vendor pricing and latency figures must be tied to specific versioned endpoints and runtime environments.

The initial falsification suite must target: every alternate credential path in the coverage inventory; state race conditions between grant validation and commit; duplicate redemption following simulated gateway crashes; policy rollback and emergency revocation during network partitions; wrong-audience and scope-expansion redemption attempts; and deliberate loss of outcome receipts following committed effects. These adversarial tests probe the fundamental assumptions underlying the strongest guarantees, rather than routine happy-path execution. A stale-state commit observed under a Level B or C adapter validates expected architectural trade-offs, not a failure of Level A guarantees.

11 Positioning among established systems

ACP represents an architectural synthesis of foundational systems concepts. Existing architectures have long separated authorization decisions from enforcement. NIST Zero Trust Architecture defines policy decision, administration, and enforcement points (PDP, PAP, PEP) [2]. Open Policy Agent (OPA) decouples declarative policy evaluation from application logic, enabling distributed sidecar enforcement [3]. Zanzibar demonstrates consistent, planet-scale relationship-based access control [11]. Classical reference-monitor principles dictate complete mediation, tamper-proof audit, and verifiable correctness [12]; object capability systems formalize least privilege, attenuated authority, and safe delegation [10]. The Model Context Protocol (MCP) standardizes tool interfaces and incorporates security boundaries [13]. These systems serve as conceptual predecessors and potential integration substrates for ACP.

The core contribution is the *explicit Evidence-to-Authority protocol* tailored to non-deterministic, agentic workloads: binding canonical action proposals, heterogeneous and potentially correlated model judgments, consequence-aware assurance obligations, cryptographic capability grants, commit-time gateway redemption, and authoritative post-state outcome receipts. A conventional PDP/PEP infrastructure that fully realizes this contract is functionally equivalent. The architectural value of ACP resides in the formal rigor of this contract, its explicit failure containment under partial model corruption, and its quantifiable operational overhead, not in claiming novelty for capabilities or control planes in the abstract.

12 Implementation and validation boundaries

The protocol above is a design specification, not evidence that a deployment satisfies its assumptions. The following checks determine which claims an implementation can support; they add no protocol stages or conformance requirements:

1. **Comparative value.** Any claimed advantage over a hardened PDP/PEP with capability tokens and verifiers needs a matched evaluation of security and operational effects.
2. **Enforcement coverage.** Verify the coverage inventory and resource-side permissions to exclude gateway bypass. Claim Level A state protection only where the commit-relevant predicate is enforced atomically; otherwise declare Level B or C and its race window.
3. **Availability and revocation.** Measure gateway replicas, short-lived grants, policy epochs, key rotation, and revocation during partitions before claiming an availability or revocation bound.
4. **Evidence and assurance.** Preserve the meaning, provenance, and question scope of each evidence type; record deterministic checks and probabilistic policy thresholds; test known witness dependencies and bound diversity claims where common dependencies are unknown.
5. **Recovery and delegation.** Report uncertain effects as unknown until reconciliation, without inferring exactly-once effects from redemption. Verify child-grant scope and revocation across the claimed delegation lineage.
6. **Decision quality.** Use an independent oracle to report protocol violations separately from semantically bad but authorized actions.

These checks bound deployment and comparative claims. Where evidence is missing, state the narrower guarantee rather than implying that the specified protocol has already delivered a measured result.

13 Glossary and protocol status

Term	Meaning in this specification
Proposal	Canonical, non-authoritative structured request for a typed protected mutation.
Evidence-to-Authority boundary	The architectural and protocol boundary that transforms verified, action-specific assurance into a narrowly scoped execution grant.
Decision provider	An independent evaluation component generating typed candidate evidence; includes LLM evaluators, deterministic verifiers, formal provers, telemetry sources, and human reviewers.
Candidate evidence	Structured, versioned evaluation result binding question schema, state view, observation timestamp, and provenance metadata; conveys zero execution authority.
Witness	A realized candidate evidence item verified and accepted to discharge a named assurance obligation.
Assurance obligation	A policy-specified predicate and witness requirement governing a canonical action under defined environmental conditions.
Admission certificate (C_q)	An integrity-protected attestation minted by the assurance service certifying that all obligations for a proposal were satisfied under verified evidence at admission time; confers no execution authority.
Execution grant (G)	A signed capability binding principal, action, resource, parameter predicate, policy version, validity window, commit-time state predicate, and single-use nonce.
Redemption	Gateway validation and durable expenditure of an execution grant at the protected mutation boundary immediately prior to dispatching an effect.
Authoritative outcome	A verified observation of execution success, failure, or unreconciled status emitted directly by the execution gateway or target resource.
Adapter guarantee level	A declared classification of state-bound execution enforcement: atomic state-bound (Level A), pre-execution revalidated (Level B), or detection and post-hoc reconciliation (Level C).
Epistemic fault domain	A shared latent dependency (e.g., training data, prompt templates, retrieval corpus, or operational infrastructure) capable of inducing correlated failures across seemingly distinct providers.

Normative intent

The uppercase message names and the grant fields are proposed interoperable protocol elements. The exact serialization, cryptographic suite, authentication profile, resource registry, policy language, and deployment topology remain to be specified. A future interoperable version would need conformance tests for canonicalization, state predicates, nonce persistence, revocation, and outcome reconciliation.

14 Design With Us

OpenKedge is developing the Agentic Control Plane as an open architecture for governing consequential autonomous operations. We collaborate with engineering and security teams deploying agents that

interact with cloud infrastructure, IAM systems, production databases, financial rails, and sensitive APIs. This document specifies the candidate control-plane protocol and the empirical validation suites required to verify its guarantees; it does not claim pre-existing enterprise deployments.

A practical adoption engagement begins with a single high-consequence operation: inventory all existing mutation paths to establish complete mediation, define the canonical action schema and assurance obligations, issue action-bound grants, declare the adapter guarantee level, and systematically execute the bypass and race test suites detailed in Section 10. Existing agent frameworks and prompt pipelines remain intact above the boundary, while mutation credentials and execution enforcement transition to dedicated gateway control.

We invite systems researchers, security practitioners, and platform architects to collaborate on gateway adapter development, protocol formalization, empirical benchmarking, and reference implementations. The most valuable contributions at this stage are concrete counterexamples, rigorous adversarial evaluations, and reference adapters that make the protocol's theoretical assumptions empirically testable.

Document status

This is a proposed architecture and protocol, not an implementation report. Formal properties are conditional on the stated trust and enforcement assumptions. No ACP prototype results or vendor performance claims are reported. OpenAI Codex assisted with structure and wording; the authors remain responsible for technical and source verification.

References

[1] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to attribute based access control (ABAC) definition and considerations. Technical Report SP 800-162, National Institute of Standards and Technology, 2014. Updated August 2019.

[2] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero trust architecture. Technical Report SP 800-207, National Institute of Standards and Technology, 2020.

[3] Open Policy Agent. Open policy agent documentation: How to deploy opa. https://www.openpolicyagent.org/docs/deploy, 2026. Accessed 2026-09-27.

[4] Open Networking Foundation. OpenFlow switch specification, version 1.5.1. Technical report, Open Networking Foundation, 2015.

[5] Jun He and Deying Yu. Persistent cognitive identity: A systems architecture for continuity across AI substrates and embodiments. OpenKedge technical manuscript. https://www.openkedge.io/paper/persistent-cognitive-identity.pdf, 2026.

[6] Jun He and Deying Yu. Cognitive admission control: Risk-conditioned assurance for consequential actions in agentic distributed systems. *arXiv preprint arXiv:2609.16313*, 2026. https://arxiv.org/abs/2609.16313.

[7] Jun He and Deying Yu. Before agents act: Assurance-aware semantic scheduling for evidence acquisition in distributed systems. Unpublished OpenKedge manuscript, 2026.

[8] Jun He and Deying Yu. The honest quorum problem: Epistemic byzantine fault tolerance for agentic infrastructure. *arXiv preprint arXiv:2607.16109*, 2026. https://arxiv.org/abs/2607.16109.

[9] Jun He and Deying Yu. Agent-native telemetry: Verifiable state-delta evidence for autonomous operations. *arXiv preprint arXiv:2608.16178*, 2026. https://arxiv.org/abs/2608.16178.

[10] Mark S. Miller, Ka-Ping Yee, and Jonathan Shapiro. Capability myths demolished. Technical Report SRL2003-02, Johns Hopkins University Systems Research Laboratory, 2003.

[11] Ruoming Pang, Ramon Caceres, Mike Burrows, Zhifeng Chen, Pratik Dave, Nathan Germer, Alexander Golynski, Kevin Graney, Nina Kang, Lea Kissner, Jeffrey L. Korn, Abhishek Parmar, Christina D. Richards, and Mengzhi Wang. Zanzibar: Google's consistent, global authorization system. In *Proceedings of the 2019 USENIX Annual Technical Conference*, 2019.

[12] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[13] Model Context Protocol. The 2026-07-28 specification. https://blog.modelcontextprotocol.io/posts/2026-07-28/, 2026. Specification release overview; accessed 2026-09-27.