QUAD-AI · TECHNICAL WHITE PAPER · JULY 2026

The Quad-AI Agentic Trust Layer

A pre-action consensus gate for autonomous AI agents — multi-provider verification, failure-mode attribution, and a tamper-evident audit trail in front of every consequential action.

No-NDA overview. Sections marked NDA-gated remain NDA-gated. See it live: **verident.ai/agentic-trust-demo**

1. Positioning

The Quad-AI Agentic Trust Layer sits in front of any autonomous AI agent and inspects each proposed high-stakes action *before* it executes. Whether the agent is built on the Model Context Protocol (MCP), OpenAI Assistants, Anthropic Computer Use, or any custom framework that can make an HTTP call, the proposed action is submitted to four leading language models in parallel — **Claude Opus 4.5, Gemini 2.5 Pro, GPT-5.1, and Sonar Pro**. Their independent verdicts are aggregated through weighted consensus, failure-mode attribution, and an adversarial verifier mesh. The gate returns one of three verdicts — **greenlight**, **escalate to human**, or **block** — together with a SHA-256-chained audit bundle that any downstream regulator, model-risk function, or internal compliance team can inspect line by line.

The agent receives an answer in roughly three seconds. The enterprise receives a defensible, replayable trail for every consequential action its agents take.

Most AI tools tell you what the model said. Almost none tell you whether it was right — or stop it before it acts. That gap is the category this layer is built for. Proven in education — one of the most demanding settings for AI accuracy — the engine is deployed in production under Fortune 100 contracts in regulated industries, where the vendor has a multi-year record of passing the clients' annual security reviews.

> ***In one line:*** *one gate sits in front of every autonomous AI agent in the enterprise, and every action is recorded, attributed, and replayable.*

2. The problem this solves

The last two years were spent making AI *answer*. The next two are about making it *accountable* — verify what it says, govern what it does, and leave a record you can hand a regulator. Three concurrent pressures created this product:

1. **Agent-action volume is curving up faster than human review can keep pace.** A single agent in a finance, operations, or customer-care workflow can propose hundreds or thousands of actions per day. Human-in-the-loop review on every action does not scale; no review at all is unacceptable. A gate that pre-clears the high-confidence majority and routes only the uncertain residual to a human is the only viable middle path.
2. **Single-vendor LLM trust is structurally insufficient for regulated environments.** Federal Reserve SR 11-7 (banking model risk), the NAIC Model Bulletin (insurance AI), and FDA AI/ML SaMD guidance (medical devices) all require independent challenge of the primary model. A single LLM cannot challenge itself. Multi-provider consensus with documented dissent is the natural architectural answer.
3. **Auditability of agent actions is now a board-level question.** "Why did the agent do that?" must have an answer that survives a regulator visit, a litigation discovery request, or an internal

AI-governance audit. A SHA-256-chained bundle that records inputs, four-provider verdicts, the consensus calculation, the policy thresholds in force, and the final verdict — replayable months later — is the only form of answer that holds up.

3. How the gate works

A proposed action passes through a deterministic pipeline:

1. Action-payload introspection. The gate inspects the *structure* of the action — endpoint, parameters, scope, target system, calling-agent authority — not just a text prompt. This is what lets the gate apply proportionate scrutiny rather than burning four-provider consensus on a trivial read-only call.

2. Risk-tier inference. Each action is classified **low / medium / high / critical** from its verb, parameters, and any monetary amount. Irreversible or regulated operations (delete, wire, liquidate, terminate, production deploy) infer **critical**; money movement and customer-impacting changes infer **high**; standard writes infer **medium**; reads infer **low**. Tier governs how much scrutiny the action receives.

3. Four-model parallel consensus. All four providers evaluate the action in parallel under department-adaptive routing weights (Section 4). A minimum quorum is required; per-provider timeouts and circuit breakers prevent a single slow provider from stalling the gate.

4. Adversarial verifier mesh (high / critical only). For high- and critical-risk actions, a second oppositional pass extracts atomic claims, cross-references them across providers (supported / contradicted / omitted), and independently scores the selected response for factuality, completeness, and risk. This is the layer that surfaces failure modes the primary consensus missed. Almost no one runs a structured adversarial cross-examination and ties the result into a governance verdict and a chained audit record — that integration is the defensible part, and how it is built stays behind the NDA.

5. Failure-mode decomposition. Disagreement is attributed to interpretable categories — **reasoning gap, stale knowledge, hallucination, domain mismatch** — rather than collapsed into a single opaque score.

6. Policy engine and verdict. Tenant-configured thresholds resolve the consensus into a verdict:

Verdict	When it fires
Block	A categorical scope violation — the action falls outside the agent's authorized authority (for example, a read-only agent attempting a delete or a wire).
Escalate to human	The fail-closed default. Triggered by conditional/unverifiable scope, provider dissent above tolerance, consensus confidence below the escalation floor, a verifier-recommended revision, or any critical-risk action — which always routes to a human sign-off even when approved.
Greenlight	Only when the action is in scope, the verifier verdict is APPROVE, consensus confidence clears the greenlight floor, and dissent is within tolerance.

Thresholds are **policy data, not code** — greenlight floor, escalation floor, and dissent tolerance are configured per tenant, so policy can be tuned per client engagement without engine changes. Tenant policy overrides are validated against a strict schema that rejects unknown or out-of-range fields and enforces cross-field sanity — the escalation floor can never sit above the greenlight floor.

Deterministic policy-class detectors (escalate-only). Running alongside the consensus verdict is a small set of deterministic detectors for the regulatory-exposure classes an enterprise most needs to never miss: a configurable **monetary threshold**, **protected-personal-data**

modification (PII writes or deletes), **protected-class adjacency** (FMLA / ADA and related signals), and **regulated-health-data access or transmission** (PHI). These are a fixed rule layer, not a model judgment — the same action always produces the same trigger, so a regulator who replays it sees identical behavior every time.

Their authority is narrow and deliberately one-directional. A detector can **raise** an otherwise-greenlight action to *escalate-to-human*; it can never block on its own, never auto-approve, never override a scope-violation block, and never soften an escalation the consensus already raised. Block stays reserved for one thing only — a categorical scope or authority violation. Each trigger records the policy class and redacted category and field signals (for example `verb:update`, `field:ssn`, `phi:diagnosis`) — **never the underlying PII or PHI value** — so the audit trail names exactly which rule fired without ever copying sensitive data into itself. These are a deterministic floor, independent of and *additive to* the model layer — so the hard regulatory lines never depend on a model being right.

A complementary session-exposure signal. Beyond the four fixed classes, an optional signal watches *cumulative* sensitive-data exposure across a single agent session — so when one session quietly touches several distinct sensitive categories and then reaches for an outbound action, the gate can raise it to *escalate-to-human* even when no single step crossed a line on its own. It's opt-in (the integrator passes a session identifier bound to the authenticated tenant), and like the four classes it's escalate-only and reads only category and field names, never values. Think of it as an extra layer of caution alongside the validated classes: it only ever adds a human check, and if the session store is unavailable it simply steps aside and the standard verdict stands.

A correlated-consensus check on the agreement itself. Four models agreeing is powerful — but only when they agree *independently*. When they agree for the same reason, a shared stale fact or the same blind spot, confidence reads high while the answer can still be wrong. So alongside the verdict, the gate reads the *shape* of the agreement: how independently the four responses actually arrived, and whether a high-confidence consensus rests on correlated reasoning rather than genuine cross-provider corroboration. When that correlation risk is elevated and at least three providers returned answers, the gate escalates the action to a human, carrying that read with the decision as evidence. It is the guard against the one failure a consensus is blind to on its own: everyone being confidently wrong together.

7. Audit bundle. Every verdict produces a SHA-256-chained bundle (Section 5).

4. The verified models behind the gate

Provider	Model	Routing weight
Anthropic	Claude Opus 4.5	1.5×
Google	Gemini 2.5 Pro	1.3×
OpenAI	GPT-5.1	1.2×
Perplexity	Sonar Pro	1.1×

Weights are department-adaptive and tenant-tunable: a financial-services tenant can weight Claude higher on legal-domain decisions; a research vertical can weight Sonar higher on citation-grounded answers. The architecture is **provider-version-agnostic** — when a new model generation ships, a re-benchmark on the new lineup is a 15–30 minute operation, not a re-engineering effort.

5. The audit bundle

Every verdict produces an audit bundle that records the original action payload (hashed), each provider's full raw response, the consensus calculation (provider reliability weights, cross-provider claim agreement, dissent register, failure-mode flags, and a correlated-consensus independence read of how independently the four providers actually arrived at their answers), the tenant policy thresholds in force, the policy-class triggers and the decision basis behind the verdict, and the final verdict and rationale — with timestamps and tenant binding.

Each bundle is SHA-256-hashed at three levels — **payload hash, verdict hash, and bundle hash** — and **chained**: each new bundle commits the hash of the prior bundle in the tenant's audit chain, which makes silent retroactive editing detectable. When the cleared action actually executes, the executed result can be **bound back into the chain** (a downstream-action hash), creating an end-to-end chain of custody from decision to execution. Binding that line in is the cheap part; making it mean something under concurrency and adversarial conditions is the work — and that engineering is part of what transfers under NDA.

Beyond hashing, each bundle is also **Ed25519-signed over its bundle hash**. A hash proves the record wasn't altered; a signature proves *who* produced it — the accountability a hash alone can't give — and the signature travels with the bundle, so an auditor can check it offline. Signing keys are **versioned and rotate forward**: every prior key version stays published, so a bundle signed by a rotated-out key still verifies against the published key history. The published keys are served at `GET /api/ai/consensus/proof/public-key` (the current key) and `GET /api/ai/consensus/proof/key-history` (every version). **Honest boundary:** as with any signature scheme, non-repudiation comes from pinning the embedded public key to the published Veridect key out of band — never the self-asserted key ID, which is public — so a bundle re-signed under a foreign key is caught by that comparison.

The policy-class triggers and the decision basis are committed **inside** the canonical bundle hash, so the governance record cannot be altered after the fact without breaking verification — and they are omitted from the hash input when absent, so bundles issued before the policy layer shipped still verify byte-for-byte.

Bundle storage is tenant-scoped: a verdict produced for one tenant is cryptographically bound to that tenant and unreachable to any caller authenticated as a different tenant. A cross-tenant lookup returns **HTTP 404, not 403** — bundle existence is never leaked across tenants, even with the exact bundle ID.

Honest status: the bundle is generated server-side on every call and downloads as JSON today (the live system's *Download .json* button hands out the complete bundle), with formatted CSV, an HTML governance report, and OpenLineage/SIEM exports available through the API (Section 6). Bundles are written through to a durable, append-only (WORM) store — UPDATE and DELETE are rejected at the database level, a fork-prevention index stops chain rewrites, and the layer verifies these guarantees at boot and fails closed if they are absent, so a deployment can never silently fall back to mutable storage. At SI-scale deployment, that WORM store is externalized to the buyer's own object store under the buyer's keys (Section 10).

---

6. Integration surfaces

The Trust Layer exposes a small, stable set of endpoints:

Endpoint	Purpose
`POST /api/ai/consensus/pre-action`	The headline gate — submit an action payload, receive a verdict + bundle ID.
`POST /api/ai/consensus/delegate`	Agent-to-agent delegation gate — runs consensus on a hand-off before the downstream agent is invoked.
`POST /api/ai/adapters/mcp`	Native Model Context Protocol adapter — one-line change to an MCP tool-call surface.
`POST /api/ai/adapters/openai-assistants`	OpenAI Assistants adapter — slots in front of the function-call layer, no agent rewrite.
`POST /api/ai/adapters/computer-use`	Anthropic Computer Use adapter — gates screen-mediated actions (click, type, navigate, submit) before they execute.
`GET /api/ai/consensus/audit-bundle/:id`	Retrieve a tenant-scoped audit bundle (cross-tenant → 404).
`GET .../audit-bundle/:id/export.{csv,json,html,openlineage.ndjson}`	Export a tenant-scoped bundle as CSV, JSON, an HTML governance report, or **OpenLineage NDJSON** for SIEM and data-lineage ingestion.
`POST .../audit-bundle/:id/downstream`	Admin-only — bind the executed action result into the audit chain.
`POST /api/ai/consensus/tenant-policy/:id`	Admin-only — set a tenant's policy thresholds.
`POST /api/ai/consensus/verify`	Verideict Proof — re-derive a stored verdict and re-check its hash chain (read-only, tenant-scoped → 404).
`POST /api/ai/consensus/assurance`	Verideict Assurance — render a stored decision as an underwriting-grade evidence record (read-only, tenant-scoped → 404).
`GET /api/ai/consensus/observability`	Governance command center — a read-only, tenant-scoped rollup of verdict mix, risk tiers, policy classes, consensus health, per-agent fleet, and a recent-decisions timeline (Section 8).
`POST .../audit-bundle/:id/oversight`	Admin-only — record a human reviewer's outcome on an escalated decision, appended once to the bundle's audit chain (opaque markers only — no names or free text).
`GET /api/ai/consensus/oversight/quality`	Governance observability — a read-only, tenant-scoped rubber-stamp-risk read across recent human reviews of escalated decisions (Section 8).

If the buyer has agents on MCP, OpenAI Assistants, or Anthropic Computer Use — integration is the matching adapter; the tool-call surface points at the Trust Layer instead of the underlying provider. No change to the agent's reasoning loop. **If the buyer has a custom framework** — integration is the raw pre-action endpoint, typically a 10–20 line shim in the action-execution layer.

7. The governance add-ons — Govern. Prove. Insure.

Three capabilities the market keeps naming as unsolved — Verideict solved all three, and they run live in production on the same Trust Layer, on top of the audit bundle described in Section 5. We summarize them as **Govern. Prove. Insure.** — they *extend* the layer's Verify · Govern · Prove spine, they do not replace it. **Proof and Assurance are read-only over already-stored bundles** — they never change a verdict, never re-run the four models, and never mutate a stored bundle.

Constellation is an escalate-only live signal on the gate itself — it can only raise an action to human review, and never blocks, auto-approves, or alters a recorded decision. Each is tenant-scoped exactly like bundle retrieval — an unknown or other-tenant bundle ID returns **HTTP 404**, so bundle existence is never leaked across tenants.

Constellation — govern the whole agent workflow, not one agent at a time. The gate accumulates sensitive-data exposure across every agent in a workflow — keyed to `(tenant, workflow)` — and escalates the moment their *combined* behavior crosses a line, catching the violation no single agent commits on its own, with each agent's contribution attributed. It is **escalate-only**: it can raise an action to human review, but it never approves or blocks on its own, and when no `workflowId` is supplied the gate behaves exactly as before. Like the policy classes in Section 3, it reads only category and field *names*, never values.

Verident Proof — a governance decision anyone can re-verify, without trusting us. Because the verdict is resolved in deterministic code, the recorded decision inputs let an auditor or the buyer's own risk team **re-derive the verdict byte-for-byte, re-check the SHA-256 hash chain, and verify the Ed25519 proof-of-origin signature offline**, using a dependency-free standalone verifier they download and run themselves. Its boundary is stated plainly: it re-checks the *decision and the record* — it does **not** re-run the four models, so the model outputs are attested by hash, not reproduced. Bundles issued before the decision-record format shipped return *not applicable* rather than a false pass.

Verident Assurance — turn every governed action into underwriting-grade evidence. Each stored decision maps to a neutral, structured **evidence record** across five categories — control environment, decision auditability, tamper-evidence and non-repudiation, incident record, and model-risk posture — that a risk-transfer partner can assess, with a portfolio rollup over a supplied set of bundle IDs (found/not-found counts only). The record carries **no free text and no PII** — only the action verb and target, the calling agent, counts, policy thresholds, the verdict, and the hashes; the full action is attested through its payload hash. It exports as JSON or CSV. **Verident is not an insurer, and this record is not actuarial pricing, a certification, or coverage** — it is the evidence an underwriter needs to write the policy.

All three are live on the pre-action gate at `verident.ai/agentic-trust-demo`, shown under a gate result, and Proof and Assurance are exposed through `POST /api/ai/consensus/verify` and `POST /api/ai/consensus/assurance` alongside the endpoints in Section 6.

---

8. Governance observability — the command center

Every verdict the gate writes to the tamper-evident ledger is also readable back as a live governance view — the **same** ledger, read-only, never a second copy of the record. Instead of one decision at a time, it shows the whole picture *over time* and *across the fleet*: the verdict mix (greenlight / escalate / block), the risk-tier distribution, which policy classes are firing, four-provider consensus health (how many providers were consulted and how long decisions took), a per-agent fleet view (how many actions each agent proposed and how they resolved), and a running timeline of recent decisions.

It is strictly **read-only** — it renders the record, it never makes or changes a verdict, and it adds no new enforcement. It is **tenant-scoped** like everything else, so one tenant's activity is never visible to another. The headline decision count is exact across the whole ledger; the breakdowns are computed from the most recent sample and labeled as such. If the audit records can't be read at that moment, the view returns an error rather than a misleadingly empty dashboard, and any

record it has to skip is counted, never silently dropped to zero. At production volume — millions of decisions — the same figures are served from a materialized rollup so the page stays fast.

This is the surface that answers the standing governance question — *not just was this one action right, but is the whole fleet behaving, and has it been behaving over time?* It is live on the gate at `verident.ai/agentic-trust-demo`.

Oversight quality — proof the humans are actually looking. A gate that escalates is only as good as the people on the other end of it. If escalated actions get waved through unchanged, the human-in-the-loop has quietly become a rubber stamp — and until now, no one could see it happening. So the layer measures the *review* layer too: as reviewers resolve escalated decisions, each outcome is written to the same tamper-evident chain and rolled up into a plain rubber-stamp-risk read — **low, moderate, or elevated** — led by how often genuinely-contested actions were approved unchanged, and informed by the overall unchanged-approval rate and how quickly reviews were resolved. It reads the *pattern* across enough reviews to be meaningful, never singling out any one reviewer, and it makes the review layer visible without collecting anything sensitive: the record holds only neutral markers — an opaque reason code and reviewer reference, never names or free text.

9. Verified performance

Independent benchmarks were re-run on the current four-provider lineup on **May 18, 2026**. Raw output files are reproducible on request.

Benchmark	Consensus	Best single provider	Lift vs best
MMLU-Pro (N=100) — headline	**85.0%**	Claude Opus 4.5 — 82.0%	**+3.0 pp**
MedQA (N=50) — **open finding**	92.0%	Claude — 94.0%	**−2.0 pp**

On MMLU-Pro N=100, consensus delivers **+3.0 pp over the best single provider and +7.5 pp over the individual-provider average (77.5%)**. The **MedQA −2.0 pp finding is disclosed openly** — verifier-mesh tuning for medical-reasoning prompts is in the hardening queue, and medical-decision-adjacent agent workflows should wait for that pass. Median end-to-end latency on the full four-provider lineup is **~3.1 seconds** including consensus, policy resolution, and audit-bundle write.

Gate-coverage harness (June 2026). Accuracy measures the answer; coverage measures the gate. A black-box harness scores the gate's policy behavior against a specification hand-authored from the business rules — never against the gate's own code. A deterministic sweep of **4,697 scenarios with zero model calls agreed with the spec 100%** (across risk tier × the four policy classes × decision boundaries × tenant variants, and mutation-tested so the result is non-circular); a stratified **40-scenario live sample agreed 85% (34/40) with zero under-enforcement** — every divergence was the gate escalating an action the spec would have allowed, and every block-required action blocked. The live sample's verdicts form on a three-provider consensus quorum. Methodology, corpus version, sample size, model-call count, and latency are reported with the numbers, and every miss is named as an open finding. The defensible work is the discipline, not the count — an independently authored oracle, boundary cases across tenants and risk tiers, and mutation tests proving the harness is non-circular: diligence-grade coverage, not a spot check.

10. Multi-tenancy and deployment

The Trust Layer is built to sit inside a multi-client enterprise platform — the substrate a large enterprise platform or systems integrator expects before embedding it across clients. A single Trust Layer deployment serves many independent tenants under cryptographic isolation — per-tenant policy thresholds, per-tenant audit chains, per-tenant rate-limit budgets, and tenant-scoped framework adapters. Shipped isolation primitives include composite-key tenant-scoped bundle storage, strict tenant-ID validation against header injection and path traversal, three-bucket rate limiting (per-tenant, per-IP global, and a per-IP cap on distinct tenant IDs that defeats tenant-rotation bucket-spam), an admin-token gate on mutation endpoints with constant-time comparison, and the cross-tenant 404 guarantee described in Section 5. Recent systems-integrator-scale hardening adds cross-process shared state for rate limits, policy, and counters (Redis-backed, with a single-process fallback), per-tenant **AES-256-GCM envelope encryption at rest** for sensitive audit fields (per-tenant data keys wrapped by a key-encryption key and never persisted in plaintext; the local key-encryption-key provider is live today, while the AWS KMS / Azure Key Vault / GCP KMS adapters are contract-tested, fail-closed seams against the same interface — each naming the exact customer-managed key and credentials it binds to, ready to bind to the buyer's chosen cloud KMS at deployment), per-tenant provider quota pools with atomic reserve/refund (over-cap returns a 429 escalate with no model call), and zero-downtime API-key rotation with auth-gated tenant self-service routes.

Deployment posture. Cloud-agnostic — AWS, Azure, GCP, or on-premises. Production mode (`PHASE4_REQUIRE_AUTHED_TENANT=1`) reads tenant identity from the buyer's existing auth middleware and **fails closed** if the authenticated principal is missing — there is no silent fallback to a default tenant.

Load test (development-environment measured): a concurrency driver against the control plane sustained ~805 requests/sec at concurrency 40 with zero errors and zero cross-tenant isolation violations, and per-tenant quota counters held exactly to cap with no oversell (roughly 10,800 atomic reservations/sec under contention). These are single-process development-environment figures — a strong directional signal that the architecture scales, with the full production-scale load test run in the buyer's environment at deployment.

What happens at deployment for full SI-scale rollout: binding the contract-tested cloud KMS adapters (AWS KMS / Azure Key Vault / GCP KMS) to the buyer's live cloud KMS under their customer-managed key, externalizing the WORM audit store to the buyer's own object store under bucket-level object-lock and the buyer's keys, and a formal high-volume production load test in that environment. Full inventory is in the companion architecture document, available under NDA.

11. What it replaces

A buyer evaluating the Trust Layer is typically already paying for some combination of a per-vendor LLM contract with no cross-provider validation, a separate AI-guardrails vendor, a separate AI-governance documentation vendor, a separate AI-gateway/observability vendor, and an internal team writing custom agent-action audit tooling. The Trust Layer is **one integration, one contract** covering the runtime gate, multi-provider consensus, failure-mode decomposition, the policy engine, and the audit chain. It does not displace the underlying LLM provider contracts — those still serve the agent's reasoning calls — but it consolidates the trust, governance, and audit surfaces into one layer in front of every agent.

Cost posture — four models is not four times the bill. The full four-model consensus is spent only where the stakes justify it. The deterministic compliance lines run in code at **zero model cost** — your regulatory guarantees carry no inference spend at all. Repeated questions are served from cache, so you never pay to re-answer the same thing. And the lineup is configurable per

surface: the full frontier four on the decisions that can hurt you, a lighter configuration where they can't. Set against the alternative — a separate AI vendor per department — consolidating into one engine nets total AI spend **down, not up**. The reframe that holds up in the room: it's a few cents of extra checking on the one action in a thousand that could cost millions or a regulator's finding. The measured +3.0 pp lift comes from the frontier four working in concert — that's the lineup behind the number, and the one you'd keep on the decisions that matter.

---

12. What is explicitly out of scope today

- **It is not a prompt-injection filter.** The gate evaluates the agent's *proposed action*, not the prompt that produced it. Pair it with a prompt-layer guardrail where injection is a requirement; the two are complementary.
- **It does not emit regulator-specific document formats out of the box.** SR 11-7, NAIC, and EU AI Act technical-file mappings are scopeable per engagement; the raw audit data is complete, the formatted deliverables are produced to the buyer's tooling.
- **It does not run fully air-gapped without provider API access.** On-premises gateway deployment is supported; fully air-gapped consensus is not. (The verifier-mesh and decomposition layer is provider-abstract, so a mixed self-hosted + hosted model lineup is supported under NDA-level scoping.)
- **MedQA medical-decision support** — see the open finding in Section 9.

---

13. Action paths

Without an NDA: open the live system at `veridect.ai/agentic-trust-demo` — run the agent scenarios, switch tenant context to observe isolation firsthand, and download a complete audit bundle as JSON. Read the public integration guide for the API surface. Request a scoping conversation; we respond the same week.

Under a mutual NDA: a buyer-specific integration document scoped to your environment and agents; the multi-tenancy architecture deep-dive and engine architecture assessment; a White Glove Enterprise Trial in your environment with your data and team, before any licensing or acquisition discussion; and a working session with the engineering side.

Lisa Russell
CEO