Quad-AI Consensus Engine — Public Integration Guide

Integration Specifications for Enterprise Buyers · Includes Agentic Trust Layer (May 2026)

Audience: Buyer-side engineering, architecture, and procurement teams evaluating Quad-AI for integration into their applications, autonomous-agent stacks, or internal AI governance posture. Sharable freely at first contact — no NDA required. NDA-gated material is called out explicitly in Section 18.

Purpose: Single-document public integration guide covering what the consensus engine does, what's new in May 2026 (Agentic Trust Layer + multi-tenant posture), integration options, full API reference (consensus + streaming + Agentic Trust Layer endpoints + health/status), response schema, code examples in Python / JavaScript / cURL, task-type routing, circuit-breaker and cache behavior, error handling, rate limits, authentication and multi-tenant production posture, security and compliance, autonomous-agent integration paths, what's NDA-gated, getting-started proof-of-concept path including the 15-minute live proof path, and the companion AI Growth Measurement Framework asset.

Classification: Public — No NDA Required · **Version 2.1 · Updated July 2026 See It Live:** verident.ai/quad-ai-demos · verident.ai/agentic-trust-demo

Table of Contents

1. What the Consensus Engine Does

The Quad-AI Consensus Engine is multi-model AI middleware. Your application sends a single API request. The engine:

1. Fires the query to 4 AI providers in true parallel (not sequentially, not as fallbacks)
2. Applies weighted consensus voting based on query type and provider reliability
3. Monitors each provider through independent circuit breakers
4. Returns the consensus-verified answer with a complete audit trail

From your application's perspective, it looks like calling one AI provider. Under the hood, four independent models verify each other before the answer is returned — and on high-stakes calls, each answer is cross-examined by the other models before any verdict is issued. 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.

What your application receives. The consensus-verified answer · a confidence score (0–100%) · per-provider latency and success/failure status · a unique trace ID for audit and replay · consensus method used · circuit breaker status for each provider · cost saved through early consensus and caching.

A note on cost. Four models does not mean four times the AI bill. You pay for the full four-model check only where the stakes justify it — the hard compliance lines are enforced in deterministic code at zero model cost, early consensus cuts spend on routine calls by 25–40%, and cache hits cost nothing. Verification lands on the decisions that can actually hurt you, not on every routine call. Governance scales with risk, not with your bill.

CURRENT PROVIDERS

Provider	Model	Weight	Strength
Anthropic	Claude Opus 4.5	1.5×	Deep reasoning, step-by-step logic
Google	Gemini 2.5 Pro	1.3×	STEM, scientific accuracy, multimodal
OpenAI	GPT-5.1	1.2×	Creative reasoning, complex reasoning chains
Perplexity	Sonar Pro	1.1×	Real-time data, fact verification with citations

2. What's New in May 2026 — Agentic Trust Layer + Multi-Tenant Posture

Two major capabilities shipped in May 2026.

Agentic Trust Layer. A pre-action consensus gate that sits in front of any autonomous AI agent built on MCP, OpenAI Assistants, or Anthropic Computer Use. Returns a greenlight / escalate / block verdict in roughly three seconds with a SHA-256-chained audit bundle for every action.

Multi-Tenancy Harness. A single deployment now serves many independent client tenants under cryptographic isolation. Per-tenant API keys, per-tenant policy thresholds, per-tenant audit chains, per-tenant rate-limit budgets. Production-mode auth (`PHASE4_REQUIRE_AUTHED_TENANT=1`) fails closed (401) — no silent fallback. Cross-tenant isolation is verifiable on the live system.

Since this revision — June and July 2026. The SI-scale hardening items open in May have shipped and been load-tested: durable write-once audit-bundle storage, cross-process rate-limit and quota state, a per-tenant API-key provisioning lifecycle, and per-tenant provider quota pools. A June 2026 load run recorded zero cross-tenant isolation violations (development-environment evidence; what remains is binding to the buyer's own key management and a formal production load test). July 2026 added an input-safety stage on the consensus Q&A path (Section 16) and an optional session-level cumulative-exposure signal on the pre-action gate (Section 17). July 2026 also shipped three governance add-ons on the pre-action gate — the three governance problems the market keeps calling unsolved, now solved and live: **Constellation** (workflow-level cumulative-exposure escalation of a live action — escalate-only, it can only add a human check), **Verident Proof** (re-derive any stored verdict and re-check its SHA-256 chain offline via a standalone verifier; `POST /api/ai/consensus/verify`), and **Verident Assurance** (render a stored decision as a neutral, PII-free five-category evidence record; `POST /api/ai/consensus/assurance`). Proof and Assurance are read-only over stored bundles — they never change a decision or re-run a model; all three are tenant-scoped (cross-tenant lookup → 404). Verident is not an insurer; the Assurance record is evidence for an underwriter, not pricing, certification, or coverage. July 2026 also shipped a read-only **governance command center** (`GET /api/ai/consensus/observability` , Section 6) — a tenant-scoped rollup over the same audit ledger showing verdict mix, risk tiers, policy-class activity, consensus health, and a per-agent fleet view over time; it only reads the record and adds no new enforcement. Also in July 2026: a **correlated-consensus signal** on the pre-action gate that reads how independently the four providers arrived at their answers and escalates a high-confidence consensus resting on correlated reasoning; and an **oversight-quality read** (`GET /api/ai/consensus/oversight/quality`) that measures how human reviewers handle escalations and surfaces any rubber-stamp pattern, so the human-in-the-loop can be shown to be genuinely engaging. Both are verified in source and covered by unit tests, and both postdate the May external validation.

3. Integration Options

Option A — Consensus-as-a-Service (API). Your application calls our hosted engine. Replace one API call. Deployed in days.

`Your App` → `POST /api/ai/quad-consensus` → `Consensus-Verified Answer + Audit Trail`

Option B — Agentic Trust Layer for Autonomous Agents. Wire your agent's tool-call surface to one of three adapters and gate every high-stakes action through the consensus engine before execution. See Section 17.

Option C — Owned Orchestration Layer. You acquire the engine and deploy it in your own infrastructure. Plug in your own AI providers (including proprietary or open-source models). Custom provider adapters available under NDA.

Option D — Regional / Portfolio License. Exclusive deployment rights for a region or across a portfolio of companies. Per-query pricing available.

4. API Reference — Consensus Endpoint

`POST /api/ai/quad-consensus` — sends a query to all 4 providers in parallel and returns the consensus-verified answer.

REQUEST HEADERS

```
Content-Type: application/json
Authorization: Bearer <your_api_key>
X-Tenant-Id: <your_tenant_id>        # required in multi-tenant production mode
X-Correlation-ID: <optional_tracking_id>
```

REQUEST BODY

```
{
  "prompt": "Explain photosynthesis to a 4th grader",
  "systemMessage": "You are a science tutor for elementary students.",
  "taskType": "explanation",
  "metadata": {
    "clientId": "your-app-id",
    "userId": "anonymous-user-hash",
    "domain": "science",
    "gradeLevel": "4"
  }
}
```

SUCCESSFUL RESPONSE (200)

```
{
  "success": true,
  "consensus": true,
  "result": "Photosynthesis is how plants make their own food...",
  "confidence": 85,
  "providersUsed": 4,
  "totalProviders": 4,
  "traceId": "quad-1716000000000-abc1234",
  "providers": [
    { "name": "Claude Opus 4.5", "model": "claude-opus-4-5", "ok": true, "latencyMs": 2847 },
    { "name": "GPT-5.1",         "model": "gpt-5.1",         "ok": true, "latencyMs": 1923 },
    { "name": "Gemini 2.5 Pro",  "model": "gemini-2.5-pro",  "ok": true, "latencyMs": 1350 },
    { "name": "Sonar Pro",       "model": "sonar-pro",       "ok": true, "latencyMs": 1612 }
  ],
  "evidence": {
    "traceId": "quad-1716000000000-abc1234",
    "parallelExecution": true,
    "consensusMethod": "weighted-voting",
    "earlyConsensus": true,
    "cacheHit": false
  }
}
```

5. API Reference — Streaming Endpoint

`POST /api/ai/quad-consensus/stream` — same request format, returns results via Server-Sent Events (SSE) as each provider completes.

```
event: provider_complete
data: {"provider":"gemini","name":"Gemini 2.5 Pro","latencyMs":1350,"ok":true}

event: provider_complete
data: {"provider":"openai","name":"GPT-5.1","latencyMs":1923,"ok":true}
```

```
event: provider_complete
data: {"provider":"anthropic","name":"Claude Opus 4.5","latencyMs":2847,"ok":true}

event: early_consensus
data: {"confidence":85,"providersComplete":3,"threshold":75}

event: consensus_result
data: {"success":true,"consensus":true,"result":"...","confidence":85,"traceId":"quad-..."}

event: done
data: [DONE]
```

6. API Reference — Agentic Trust Layer Endpoints

Twelve endpoints in production, all under `/api/ai/`.

Endpoint	Method	Purpose
`/consensus/pre-action`	POST	Pre-action gate — accepts an action payload, returns greenlight / escalate / block verdict + bundleId
`/consensus/delegate`	POST	Agent-to-agent delegation gate
`/consensus/audit-bundle/:bundleId`	GET	Retrieve SHA-256-chained audit bundle for a prior decision (tenant-scoped — cross-tenant lookup returns 404)
`/adapters/mcp`	POST	MCP-shaped adapter for any MCP-capable agent
`/adapters/openai-assistants`	POST	OpenAI Assistants-shaped adapter (slots in front of the function-call layer)
`/adapters/computer-use`	POST	Anthropic Computer Use-shaped adapter
`/consensus/tenant-policy/:tenantId`	POST	Per-tenant policy management (admin-token gated, constant-time compare)
`/consensus/verify`	POST	Veridect Proof — re-derive a stored verdict and re-check its SHA-256 chain offline (read-only; returns *not applicable* for bundles predating the decision-record format)
`/consensus/assurance`	POST	Veridect Assurance — render a stored decision as a neutral, PII-free five-category evidence record (read-only; not insurance pricing, certification, or coverage)
`/consensus/observability`	GET	Governance command center — read-only, tenant-scoped rollup of verdict mix, risk tiers, policy classes, consensus health, per-agent fleet, and a recent-decisions timeline (reads the audit ledger back; adds no new enforcement)
`/consensus/audit-bundle/:bundleId/oversight`	POST	Record a human reviewer's outcome on an escalated decision — admin-token gated, appended once to the bundle's audit chain (opaque markers only, never names or free text)
`/consensus/oversight/quality`	GET	Oversight-quality read — read-only, tenant-scoped rubber-stamp-risk rollup across recent human reviews of escalated decisions

PRE-ACTION GATE — REQUEST

```
{
  "tenantId": "tenant-abc",
  "agentId": "agent-finance-bot-1",
  "actionType": "external_api_call",
  "actionPayload": {
    "endpoint": "https://api.bank.example.com/transfer",
    "amount": 5000,
    "currency": "USD",
    "destination": "account-xyz"
  },
  "riskTier": "high",
  "context": "Customer-initiated wire transfer"
}
```

PRE-ACTION GATE — RESPONSE

```
{
  "verdict": "escalate",
  "confidence": 72,
  "bundleId": "bundle-1716000000000-xyz789",
  "consensus": {
    "providersAgreed": 3,
    "totalProviders": 4,
    "method": "weighted-voting + verifier-mesh"
  },
  "policyApplied": {
    "tenantId": "tenant-abc",
    "threshold": 80,
    "escalationPath": "human-review-queue-1"
  },
  "rationale": "Amount exceeds tenant-policy escalation threshold of $1,000. Two providers flagged destination risk.",
  "auditChain": {
    "previousBundleHash": "sha256:abc123...",
    "thisBundleHash": "sha256:def456..."
  }
}
```

Verdicts are strict: **greenlight** (proceed) · **escalate** (human review required) · **block** (do not execute). The `bundleId` is the canonical reference for replay and regulator review.

Every decision — and, when the caller reports it, the real action executed after a greenlight — is bound into the same tamper-evident chain, so an auditor can independently replay and verify what was authorized and what was actually done. The binding itself is one line of integration; the engineering that makes it trustworthy under concurrency and adversarial conditions is part of the NDA-gated build.

6.5 Externally Verified Behavior on the Live System

Within 72 hours of the Agentic Trust Layer shipping on May 19, 2026, a credentialed Fortune 100 model-risk reviewer ran seven adversarial scenarios against the live public agentic-trust surface. The pre-action gate, the four-provider consensus, the per-claim heatmap, the failure-mode decomposition, and the SHA-256-chained audit bundle were all exercised on real agent tool-call payloads. **Every scenario produced behavior matching the pre-stated expectations described to the reviewer in writing before he ran them.**

Externally verified deterministic gate-policy detectors (escalate-only — below the configured threshold the action is not flagged and the overall gate greenlights; at or above it the detector fires escalate, with the exact triggered rules named in the audit bundle):

- **$-threshold** — at $99k the detector does not fire (overall gate greenlight); at $101k it fires escalate
- **PII-modify** — no PII modification does not fire (overall gate greenlight); a PII modification fires escalate
- **Protected-class adjacency (FMLA / ADA / NLRA overlap)** — fires escalate on the HR policy edge case with retaliation risk
- **Regulated health-data access + transmission (PHI)** — fires escalate on the cross-specialty agentic referral; the audit bundle named HIPAA Treatment Exception + Minimum Necessary by name (external-party disclosure and minimum-necessary appear as named HIPAA rule context in the bundle, not as separate detectors)

These deterministic detectors are escalate-only — they never greenlight, block, override a block, or downgrade a verdict. Block remains reserved for no-authority scope violations. Trigger evidence records the policy class and named category and field signals (for example `verb:update`, `field:ssn`, `phi:diagnosis`) — never the raw PII/PHI value.

The High risk tier was externally observed live for the first time on the PHI scenario, confirming risk-tier inference is wired correctly (not just policy-class triggering). The fast-path tier is confirmed wired and not collapsing to "escalate everything." The engine-knows-its-lane meta-property (refusing to cross into "legal advice" and routing to humans) was externally verified across three regulated domains: legal, HR, and healthcare HIPAA.

Full per-scenario instrumentation, the four open findings named openly at the time (MedQA −2.0pp; formatted compliance-tool adapters partial; multi-tenancy SI-scale hardening then in progress — since shipped and load-tested, June 2026; true air-gap requires self-hosted substitution), and the reusable verbatim reviewer quotes are documented

in the External Validation & Testing Record, available under NDA. The reviewer is anonymous and not citable by name without permission; the validation results and quotes are reusable.

7. API Reference — Health & Status

- `GET /ready` — returns 200 if the engine is ready (load balancer health check)
- `GET /alive` — returns 200 if the engine process is running (container liveness probe)
- `GET /metrics/prometheus` — Prometheus-format metrics
- `GET /api/ai/system-status` — aggregated system health, provider availability, circuit breaker state, cache hit rates

8. Response Schema — Complete Decision Record

Every response from the consensus engine includes a complete decision record designed for regulatory compliance, audit, and debugging.

Top-level fields: `success` · `consensus` · `result` · `confidence` (0–100; 75+ = high confidence) · `providersUsed` · `totalProviders` (always 4) · `traceId` (format `quad-{timestamp}-{random}`).

Provider array: for each provider — `name` · `model` · `ok` · `latencyMs` · `response` (first 100 chars; full responses under NDA) · `error` (sanitized).

Evidence object (audit trail): `traceId` · `parallelExecution` · `startTime` · `endTime` · `totalLatencyMs` · `consensusMethod` (`"weighted-voting"` / `"cache"` / `"none"`) · `strictMode` · `earlyConsensus` · `abortedProviders[]` · `cacheHit` · `costSaved`.

Agentic Trust Layer additions on agentic-trust responses: `verdict` · `bundleId` · `policyApplied` · `rationale` · `auditChain.previousBundleHash` · `auditChain.thisBundleHash` · `attestation` (Ed25519 proof-of-origin: `alg` = `ed25519` · `keyId` · `publicKey` · `signedField` = `bundleHash` · `signature` · `keyVersion` (optional; absent on pre-signing bundles, treated as version 1)).

Every audit bundle generated by an Agentic Trust Layer verdict can be retrieved by `bundleId` via `GET /api/ai/consensus/audit-bundle/:bundleId` (tenant-scoped — cross-tenant lookup returns HTTP 404, not 403, so existence is not leaked across tenants), or downloaded as a JSON file directly from the live system via the **Download .json** button on `/agentic-trust-demo`. The bundle contents include each provider's full raw response, the routing weights applied at decision time, the per-claim consensus heatmap, the four-failure-mode decomposition, the weighted synthesis, the consensus confidence score, and the SHA-256 hash chain across the bundle. Each bundle is additionally **Ed25519-signed over its bundle hash** for proof-of-origin; the current published key is served at `GET /api/ai/consensus/proof/public-key` and every signing-key version at `GET /api/ai/consensus/proof/key-history`, so a bundle signed by a rotated-out key still verifies against the published key history. Non-repudiation comes from pinning the embedded public key to the published Veridect key rather than the self-asserted key ID.

9. Code Examples

PYTHON — CONSENSUS

```
import requests

QUAD_AI_URL = "https://your-endpoint.com/api/ai/quad-consensus"
API_KEY = "your_api_key"
TENANT_ID = "your_tenant_id"

def get_consensus(prompt, task_type="question", system_message=None):
    payload = {"prompt": prompt, "taskType": task_type}
    if system_message:
        payload["systemMessage"] = system_message
    r = requests.post(QUAD_AI_URL, json=payload, headers={
        "Content-Type": "application/json",
        "Authorization": f"Bearer {API_KEY}",
        "X-Tenant-Id": TENANT_ID,
    }, timeout=60)
    r.raise_for_status()
    data = r.json()
```

```
    if data["consensus"]:
        return data["result"]
    return None  # escalate to human review
```

PYTHON — PRE-ACTION GATE FOR AN AUTONOMOUS AGENT

```
def gate_action(agent_id, action_type, payload, risk_tier="medium"):
    r = requests.post(
        "https://your-endpoint.com/api/ai/consensus/pre-action",
        json={
            "tenantId": TENANT_ID, "agentId": agent_id,
            "actionType": action_type, "actionPayload": payload,
            "riskTier": risk_tier,
        },
        headers={"Authorization": f"Bearer {API_KEY}", "X-Tenant-Id": TENANT_ID},
        timeout=30,
    )
    r.raise_for_status()
    return r.json()  # contains verdict + bundleId

verdict = gate_action("agent-1", "external_api_call",
                      {"endpoint": "...", "amount": 5000})
if verdict["verdict"] == "greenlight":
    execute_action()
elif verdict["verdict"] == "escalate":
    queue_for_human_review(verdict["bundleId"])
else:
    log_block(verdict["bundleId"], verdict["rationale"])
```

JAVASCRIPT / NODE.JS — CONSENSUS

```
const r = await fetch("https://your-endpoint.com/api/ai/quad-consensus", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Authorization": `Bearer ${API_KEY}`,
    "X-Tenant-Id": TENANT_ID,
  },
  body: JSON.stringify({ prompt, taskType: "question" }),
});
const data = await r.json();
if (data.consensus) return data.result;
```

CURL

```
curl -X POST https://your-endpoint.com/api/ai/quad-consensus \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer your_api_key" \
  -H "X-Tenant-Id: your_tenant_id" \
  -d '{"prompt":"What is the capital of France?","taskType":"question"}'
```

---

10. Task Types & System Message Routing

When you provide a `taskType` instead of a custom `systemMessage`, the engine selects an optimized system prompt and may adjust routing weights.

Task Type	Description
`question`	Direct factual questions — optimized for accuracy and conciseness
`explanation`	Detailed concept explanations

Task Type	Description
`analysis`	Content analysis, pattern recognition, insights
`homework`	Guided problem-solving (teaches, doesn't just answer)
`assessment`	Evaluate work, provide constructive feedback
`creative`	Generate creative content

Custom system messages override task-type defaults. Use custom messages when your domain (healthcare, legal, finance) requires specific instructions.

11. Circuit Breaker Behavior

Each provider has an independent circuit breaker — failure threshold 3 consecutive failures · 30-second reset · 2 half-open probe requests · per-provider request timeout 25–40s.

State	What Happens	Your App Sees
CLOSED	Normal operation	Provider returns with `ok: true`
OPEN	Provider blocked	Provider returns `ok: false`, `error: "temporarily unavailable"`
HALF-OPEN	Testing recovery	May return `ok: true` or `ok: false`

The engine handles retries automatically. You do not need retry logic. Degradation is automatic — 4/4 → 3/4 → 2/4 → escalation.

12. Cache Behavior

Scenario	Latency	Cost
Cache miss	2.2–3.5 s	Full API cost
Cache hit	< 50 ms	Zero
Early consensus (3/4)	1.5–2.8 s	25–40% savings

Key generation: SHA-256 of normalized prompt + model + temperature + maxTokens. Default TTL 24h; time-sensitive TTL 1h. Hybrid Redis + in-memory LRU.

13. Error Handling & Graceful Degradation

The engine is designed to never fail silently. Every error state is explicit.

Status	Meaning	Your Action
200	Success — check `consensus` field for result quality	Parse normally
400	Bad request	Fix payload
401	Invalid or missing API key, or missing required tenant binding in production mode	Check headers
404	Audit bundle requested by wrong tenant — no existence leak	Verify tenant ID
429	Rate limit exceeded	Back off and retry
500	Internal error	Retry once; if persistent, contact support with `traceId`

Providers Available	Behavior	Confidence
4/4	Full consensus voting	Up to 100%
3/4	Consensus still possible	Up to 75%
2/4	Reduced consensus	Up to 50%
1/4	Single provider — no verification	25% — flagged

Providers Available	Behavior	Confidence
0/4	Engine returns error	0% — immediate escalation

14. Rate Limits & Quotas

Plan	Queries/Minute	Queries/Day	Concurrent
Proof of Concept	10	500	2
Standard	60	10,000	10
Enterprise	Custom	Custom	Custom

Three-bucket rate limiting in multi-tenant mode: per-(tenant, IP) · per-IP global · per-IP tenant-cardinality cap (defeats tenant-rotation bucket-spam).

Rate-limit headers included in every response: `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset`.

15. Authentication & Multi-Tenant Production Posture

Single-tenant mode (development / pilot). API key auth via `Authorization: Bearer <key>`. Sufficient for proof-of-concept work.

Multi-tenant production mode. When the `PHASE4_REQUIRE_AUTHED_TENANT=1` flag is set, the engine activates authenticated-tenant binding via the buyer's auth middleware (mTLS, JWT, or opaque API key). Every request must carry both `Authorization` and an `X-Tenant-Id` header. Production mode fails closed — any request without a verified tenant principal returns 401. No silent fallback. The header-tenant wins over any body-supplied tenant value to prevent bucket-split attacks.

Tenant isolation guarantees.

- Per-tenant API keys, policy thresholds, audit chains, rate-limit budgets
- Composite-key tenant-scoped bundle store — cross-tenant lookups return 404 (no existence leak)
- Strict tenant ID validation against header injection and path traversal
- Admin-token gate on mutation endpoints with constant-time compare
- Tenant-scoped framework adapters (MCP, OpenAI Assistants, Computer Use)

Multi-tenancy architecture deep-dive available under NDA.

16. Security & Compliance

Data handling. Queries and responses processed in-memory; not stored permanently unless caching is enabled. Cache entries encrypted at rest and expire automatically. No query content used for model training. All provider API calls use HTTPS / TLS 1.3.

Input safety (shipped July 2026). Inbound prompts on the consensus Q&A path clear a privacy screen before any model call: formatted personal identifiers (structured patterns such as email, credit-card, SSN, phone, and street-address formats) are scrubbed from what is stored, and only the identifier *type* and a *count* are retained — never the value. This stage fails closed: if scrubbing cannot complete, the request is rejected before a single provider is called, so sensitive input is minimized before it ever reaches a model. A content-moderation and prompt-injection check also runs ahead of the paid model calls and returns HTTP 422 when it trips — unsafe or malformed input is stopped before it reaches a provider, at zero model cost.

Compliance posture. COPPA compliant · FERPA compliant · WCAG 2.1 AA · Student Privacy Pledge 2020 signatory · SOC inheritance via DataBank (SOC 1 Type 2 + SOC 2 + CSA STAR Type 2 + SOC 3, all current 2025 with bridge letters) · HIPAA Business Associate status under executed BAAs · multi-year track record passing annual vendor security risk assessments administered by Fortune 100 vendor risk teams.

Audit trail. Every query produces a complete, immutable decision record — unique trace ID for replay · per-provider inputs, outputs, latency, status · consensus method and confidence calculation · circuit breaker state at time of query · correlation ID linking to your application's request chain. Agentic Trust Layer bundles are SHA-256-chained — each bundle hash includes the previous bundle's hash, producing a tamper-evident ledger per tenant. The chain is built over a canonical form of each record, so cosmetic reordering cannot silently alter a hash — one of several engineering details whose full design is NDA-gated.

Input validation. Prompt 1–10,000 chars · system message 1–5,000 chars · options validated against strict Zod schema · sanitized error messages (no internal details exposed).

17. Integrating an Autonomous Agent

The provider-abstraction integrity property. The claim-extraction and verifier-mesh layer is provider-abstract by design. Any provider — Claude, GPT, Gemini, Sonar, or a self-hosted Llama / Mistral / Qwen / open-weight substitute — that conforms to the abstraction layer's response contract (raw response + confidence signals + structured claim list) plugs in. The heatmap, the failure-mode decomposition, the consensus synthesis, and the SHA-256-chained audit bundle continue to work identically over the mixed lineup. A buyer running a mixed Llama + Claude + GPT + Sonar configuration in an on-prem or air-gapped environment receives the same governance surface as a buyer running pure hosted quad on SaaS. This property was independently confirmed by the May 19–20, 2026 external reviewer as "the non-negotiable you should treat as sacred. Keeping the exact same claim-extraction + verifier-mesh + heatmap/decomposition surface across mixed hosted/self-hosted providers is what makes the whole engine valuable in regulated or air-gapped environments."

The same property is a cost lever: a model upgrade — or a swap to a cheaper model generation on a lower-stakes surface — is a configuration change, not a rebuild.

Three integration paths, in increasing order of customization.

Path 1 — MCP Adapter. Point the agent's tool-call surface at `/api/ai/adapters/mcp`. One-line change. Every tool call is gated through the consensus engine before execution.

Path 2 — OpenAI Assistants Adapter. Same shape, at `/api/ai/adapters/openai-assistants`. Slots in front of the function-call layer with no agent-code rewrite — the adapter speaks the Assistants tool-call protocol natively.

Path 3 — Raw Pre-Action Endpoint. For custom agent frameworks, wrap each high-stakes action in a single HTTP call to `/api/ai/consensus/pre-action` before execution. Typically a 10–20 line shim.

For Anthropic Computer Use specifically, use `/api/ai/adapters/computer-use` — the adapter intercepts the action loop and gates each screen-altering action.

Optional session-level exposure signal (shipped July 2026). When your integration passes a session identifier (bound to the authenticated tenant), the gate also watches *cumulative* sensitive-data exposure across that session and can escalate an outbound action once several distinct sensitive categories have been touched — even when no single step crossed a line alone. This catches the slow, multi-step data-gathering pattern that per-action checks structurally miss. The signal is escalate-only: it can only ever add a human check, never remove one, and it steps aside if the session store is unavailable. It records category names only — never values.

Sandbox endpoint with buyer-specific tenant ID, custom policy thresholds, and engineering walkthrough available by our second working session after mutual NDA.

18. What's Included Under NDA

Document	Contents
Custom Integration Package	Company-specific API configuration, dedicated endpoints, custom routing weight profiles
Provider Adapter Specification	How to add your own AI models (proprietary, open-source, regional)
Routing Weight Methodology	How domain-specific weights are calibrated; testing methodology; failure patterns per provider
Circuit Breaker Deep Dive	Full state machine, failure pattern database, recovery strategies
Cache Architecture	Key generation algorithm, TTL strategies, invalidation rules
Audit Trail Schema	Complete database schema for long-term audit storage, query replay, compliance reporting
Multi-Tenancy Architecture	Full tenant-isolation design, composite-key bundle store, rate-limit bucket math, fail-closed auth
Agentic Trust Layer Build Spec	Pre-action gate path, policy engine, three framework adapters
Architecture Assessment	10 independent AI reviews of the consensus engine architecture
Multi-Provider Strategic Analysis	10 AI providers analyze 6 industries — 60 strategic analyses
Data Architecture Pack	Full technical specification verified from source code

Document	Contents
Deployment Runbook	On-premise deployment guide, infrastructure requirements, scaling configuration
External Validation & Testing Record	Per-scenario instrumentation, the open findings named openly, reusable verbatim reviewer quotes

19. Getting Started — Proof of Concept

Step 1 — Request POC access. Contact us with your company name, primary use case, estimated query volume, and preferred integration language (Python, JavaScript, Java, C#).

Step 2 — First call.

```
curl -X POST https://your-endpoint.com/api/ai/quad-consensus \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer your_poc_key" \
  -H "X-Tenant-Id: your_tenant_id" \
  -d '{"prompt":"What is 2 + 2?","taskType":"question"}'
```

Step 3 — Replace your existing AI call. Wherever your application currently calls a single provider, swap the URL and add the tenant header.

Before:

```
response = openai.chat.completions.create(
    model="gpt-5.1",
    messages=[{"role": "user", "content": prompt}])
answer = response.choices[0].message.content
```

After:

```
r = requests.post("https://your-endpoint.com/api/ai/quad-consensus",
                  json={"prompt": prompt, "taskType": "question"},
                  headers={"Authorization": "Bearer your_api_key",
                           "X-Tenant-Id": "your_tenant_id"})
data = r.json()
answer = data["result"]        # Consensus-verified by 4 providers
trace_id = data["traceId"]     # Full audit trail
confidence = data["confidence"]
```

Step 4 — Evaluate. Compare accuracy vs your current single-provider answers · typical latency 2.2–3.5 s · audit quality (trace IDs, per-provider breakdown, confidence scores) · circuit breaker behavior during provider outages.

Step 5 — Scale or acquire. Continue as API service (per-query pricing) · license for your region or portfolio · acquire the engine and deploy in your own infrastructure.

15-MINUTE LIVE PROOF PATH (NO NDA REQUIRED)

For technical evaluators or business stakeholders who want a fast first proof point before scoping a full integration, the live public system at **verideict.ai/agentic-trust-demo** is a complete end-to-end view of the engine in production. The demo includes pre-built adversarial scenarios across financial, customer-care, operational, HR-policy, and healthcare-PHI action types. Each scenario runs the four-provider consensus, surfaces the per-claim heatmap and the four-failure-mode decomposition, returns the gate verdict (greenlight / escalate / block), and produces the SHA-256-chained audit bundle. The audit bundle can be downloaded as a JSON file via the **Download .json** button. The cross-tenant isolation guarantee is independently probable via the **Run Cross-Tenant Isolation Probe** button. End-to-end first proof: roughly 15 minutes.

20. Companion IP Asset: AI Growth Measurement Framework

A complementary IP asset offered alongside the Quad-AI Consensus Engine — bundled with the engine, bundled with the full platform, or available as a standalone acquisition.

The Problem. Traditional assessment measures static knowledge. It cannot measure what AI-augmented learners actually develop. As AI becomes ubiquitous, the entire measurement industry needs a new framework — and there isn't

one.

7 Measurement Dimensions. Problem Decomposition · Critical AI Literacy · Cross-Subject Transfer · Self-Directed Learning · Confidence Trajectory · Question Quality · Verification Instinct.

Live Implementation. veridect.ai/ai-growth-measurement — 23 pre-loaded students, classroom-level cohort analysis, per-student radar charts. Backend APIs at `/api/ai/growth-analysis` and `/api/ai/simulate-student`. Framework dossier available under separate cover.

Lisa Russell / CEO, Veridect · veridect.ai · lrussell@veridect.ai

This document contains non-proprietary integration specifications for the Quad-AI Consensus Engine and the Agentic Trust Layer. Proprietary routing weights, failure-pattern databases, multi-tenancy isolation internals, and provider adapter specifications are available under NDA.

Quad-AI · Public Integration Guide · Version 2.1