Nick Clark · Published April 25, 2026 · PDF

How to Add Governed AI to a Financial-Services Back Office

You want to run an autonomous agent against trades, positions, and risk policy, but every action it takes carries monetary loss, regulatory penalty, and systemic risk. This guide describes an architecture that decides whether each action is permitted, gated, or suspended before it commits, so that agent authority tracks market conditions and policy in real time. It is an architecture disclosed in United States Patent Application 19/647,395, not a shipping library. The home inventive step is the Applications inventive step: composing a fixed set of cognition primitives into a full-stack governed deployment for a specific regulated sector.

What You Are Building

You are building the governance layer that sits between an autonomous agent and a financial back office: order management, position keeping, risk policy, and the regulatory audit trail. The agent proposes actions. The layer decides, for every one of them, whether to permit it, gate it pending more evaluation, or suspend the agent into a non-committing mode where it can keep reasoning but cannot act.

This is the problem behind the search "how to add governed AI to a financial-services back office." A raw large language model or a raw trading policy engine will happily emit an action. What a regulated shop needs is the opposite reflex: an action does not

commit until a governance decision has explicitly cleared it, and every clearance is recorded in a form a regulator can trace after the fact.

The audience is whoever owns that boundary: a quant platform team, a risk-technology group, or an engineer standing up an autonomous or semi-autonomous execution system that must answer to position limits, value-at-risk thresholds, counterparty credit limits, and jurisdiction-specific market rules. The approach here is drawn from a domain worked example in the filed disclosure, in which a fixed set of platform primitives is applied to financial services, autonomous trading, and risk management.

Why the Obvious Approaches Fall Short

The common approaches are all reasonable and all leave a structural gap.

Pre-trade risk checks. Most firms already run limit checks in front of the order gateway: a rejection if a proposed order would breach a position or concentration limit. This is necessary but it is a pass/fail filter on individual orders. It does not model whether the agent as a whole should still be trading at all, and it does not distinguish "the market moved outside the parameters this model was validated for" from "this specific order is one lot too large."

Prompt and policy guardrails on the model. Wrapping the model in instructions or a rules layer constrains what it tends to produce. But guardrails that live inside the reasoning process can be reasoned around, and more importantly they conflate two different questions: how the agent thinks and whether the agent is permitted to act. The filed disclosure is explicit that these must be kept separate and that the second question must not depend on the first.

Log everything and reconcile later. Comprehensive logging gives you forensics, but a log written after commitment treats the committed action as a settled fact you are merely annotating. The disclosure draws the opposite line: the governance decision is

recorded as the action is authorized, as a sealed governance event, not reconstructed afterward.

The structural gap in all three is that authorization is treated as a property of the single action, computed at the moment of the order, using inputs that do not include the agent's continuously changing fitness to be trading. The disclosed architecture closes that gap by computing authorization continuously and from structured, domain-specific inputs, and by making commitment downstream of that computation rather than the other way around.

The Architecture

The disclosed approach does not introduce a bespoke financial engine. It applies a fixed set of platform primitives to the financial domain by parameterizing each one with financial inputs. This composition is the Applications inventive step. Five primitives carry the load.

1. Confidence-governed trading suspension. A confidence governor computes, continuously, whether the agent is fit to trade, from structured inputs the disclosure names explicitly: market volatility assessment (are conditions inside the system's validated operating parameters), model reliability assessment (are the predictive models producing outputs consistent with their historical accuracy distribution), data integrity assessment (are the market data feeds complete, timely, and internally consistent), position risk assessment (is current exposure within policy risk limits), and regulatory compliance assessment (do contemplated actions comply with applicable rules).

When confidence falls, suspension is graduated across three levels described in the disclosure. At the first level the system halts new position initiation but still permits management of existing positions. At the second it halts all discretionary trading and

begins orderly position reduction. At the third it transfers position-management authority to human traders and enters observation-only mode. The governor continuously re-evaluates whether to escalate or de-escalate between levels.

2. Integrity-tracked risk policy compliance. An integrity engine monitors adherence to risk policy: position limits, concentration limits, value-at-risk thresholds, and counterparty exposure limits. Each deviation, a position over a limit or a trade violating a constraint, is recorded as an integrity deviation with full semantic context. A redemption engine generates restorative actions: reducing a position back within limits, enhancing monitoring of the violated constraint, and submitting the deviation to the compliance audit trail.

3. Cryptographically sealed decision lineage. Every trading decision, risk assessment, position change, and policy evaluation is recorded as a cryptographically sealed governance event. The disclosure frames this as the mechanism for regulatory accountability: any trading action can be traced back to the specific market conditions, risk assessments, confidence evaluations, and policy evaluations that produced it.

4. Financial capability envelope. Separately from confidence, a capability envelope defines what the system is structurally authorized to do: position limits (maximum notional per instrument, sector, and aggregate), instrument eligibility, counterparty authorization and credit limits, temporal authorization (trading hours, settlement windows, execution deadlines), and regulatory authorization (jurisdiction-specific constraints). The envelope is computed continuously and, per the disclosure, feeds into the confidence governor, so authorization reflects the system's actual structural ability to execute within regulatory and risk boundaries.

5. Sandboxed affect with preserved urgency. The agent carries an affective state that modulates how it deliberates. In the financial instantiation this is deliberately bounded: risk sensitivity, novelty appetite, and persistence-under-partial-failure are

held in narrow ranges to prevent loss-aversion bias, revenge trading, or over-confidence after a winning streak. Escalation-under-time-pressure stays active, because an approaching close or a fast-moving price genuinely warrants urgency.

The load-bearing invariant across all five is a separation the disclosure enforces structurally: the governance gate evaluates admissibility from policy compliance, provenance, and trust-slope validation independently of the agent's affective state. Affective state is not an input to the gate. Even at maximal confidence disposition and minimal risk sensitivity, the agent still cannot act unless the gate independently clears it. Disposition changes how the agent thinks, never whether it is permitted to act.

How to Approach the Build

Implement this yourself, in your stack. The steps below are the order the architecture implies. The interface sketch is illustrative and faithful to the disclosure, not a package you can install.

Step 1: Put a decision boundary in front of commitment. Route every agent-proposed action through one point that returns a decision before anything reaches your order gateway or position store. The disclosure's three outcomes are the whole vocabulary of that boundary.

```
// illustrative interface, you implement it
enum Decision { PERMIT, GATE, SUSPEND }

Decision govern(ProposedAction a, MarketState m):
    envelope   = computeCapabilityEnvelope(m)    // structural authorization
    confidence = computeConfidence(m, envelope)  // fitness to trade
    if not gatePasses(a, policy, provenance):    // independent of dispositi
        return SUSPEND or GATE per level
    return PERMIT
```

Step 2: Define the confidence inputs as first-class signals. Build the five assessments as measured quantities, not vibes: volatility versus validated operating range, live model reliability versus historical accuracy distribution, data-feed integrity, position risk versus limits, and a regulatory-compliance check. Confidence is a function of these; do not let the model self-report it.

Step 3: Wire graduated suspension, not a kill switch. Encode the three levels with their own thresholds and the continuous re-evaluation loop. The important property is that suspension throttles authority progressively (new positions, then discretionary trading, then human handover) and can de-escalate when conditions recover.

Step 4: Model the capability envelope explicitly and feed it into confidence. Position, instrument, counterparty, temporal, and regulatory authorization are state, recomputed as conditions change. Do not merge this with the confidence computation; the disclosure keeps the envelope as its own primitive that then informs the governor.

Step 5: Make lineage a sealed write on the authorization path, not a downstream log. Record the governance event, the inputs that produced the decision, at the moment of authorization, cryptographically sealed. This is what makes any later action traceable to the conditions that produced it.

Step 6: Bound the affective state and keep it out of the gate. Enforce the narrow ranges on the disposition fields, keep escalation-under-time-pressure live, and structurally exclude disposition from the admissibility gate so no confidence high can ever unlock an action policy forbids.

Step 7: Choose a deployment configuration. The disclosure describes three: embedded (the governance substrate in-process with the agent behind a function-call boundary, lowest latency), co-resident (stronger isolation across a process boundary),

and hardware-assisted (lineage and cryptographic operations in dedicated hardware for the highest tamper-resistance). Latency-sensitive execution and high-assurance audit pull in different directions; pick per system.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing here "just works" out of the box. You implement every primitive against your own market-data feeds, risk system, order gateway, and compliance store, and you own the thresholds, the validated operating ranges, and the cryptographic and key-management choices behind the sealed lineage.

It is not benchmarked or production-proven here. The disclosure describes structure and mechanism; it does not supply latency figures, accuracy numbers, or a guarantee of regulatory sufficiency in any jurisdiction. Whether your instantiation satisfies a specific regulator is a question for your compliance and legal functions, not something the architecture asserts.

The governance layer is only as good as its inputs. If your model-reliability or data-integrity assessments are weak, confidence will be miscalibrated and suspension will fire late or spuriously. And the approach governs an agent's proposed actions at the commitment boundary; it does not repair an upstream trading model, and it does not remove the need for the conventional pre-trade controls it sits alongside.

Disclosure Scope

The architecture described here is disclosed in United States Patent Application 19/647,395, which applies a fixed set of cognition platform primitives (a confidence governor, an integrity engine, a capability envelope, sandboxed affective state, and cryptographically sealed decision lineage) to financial services, autonomous trading, and risk management. This guide is educational: it explains an approach a developer

can build themselves. It is not a warranty, not a claim of regulatory compliance, and not an offer of software or a downloadable implementation. Every mechanism attributed to the approach above traces to that filing; the thresholds, integrations, and assurances of any real deployment are the implementer's responsibility.

Applications (/applications)

All 40 steps → (/inventive-steps)

Same primitives. Different domains. One architecture.

Chapter 13 (/patents/19-647395/chapters/applications)

PRIMARY TECHNICAL DISCLOSURE

- One Architecture, Every Domain: How the Same Cognitive Primitives Parameterize Across Autonomous Vehicles, Defense, Companion AI, and Therapeutic Agents (/articles/domain-parameterization-one-architecture-across-autonomous-vehicles-defense-companion-ai-and-therapeutic-agents)

SECONDARY TECHNICAL

- Confidence-Governed Autonomous Driving Decisions (/articles/applications/confidence-governed-driving)
- Quorum-Based Engagement Authorization for Defense Systems (/articles/applications/quorum-engagement)
- Narrative Unlock Engine and Relationship Milestones for Companion AI (/articles/applications/narrative-unlock)
- Attachment Challenge Module: Testing Relational Health (/articles/applications/attachment-challenge)
- Skill-Gated Relational Readiness for Social Platforms (/articles/applications/skill-gated-matching)
- Fleet-Level Affective State Aggregation for Traffic Management (/articles/applications/fleet-affective)
- Therapeutic Relationship Integrity for AI-Assisted Therapy (/articles/applications/therapeutic-integrity)
- Physical Capability Envelopes for Embodied Robotics (/articles/applications/physical-capability)
- Curriculum-Gated Adaptive Learning Platforms (/articles/applications/curriculum-gated-learning)

- Continuity-Based Facility Access Control (/articles/applications/continuity-facility-access)
- Confidence-Governed Financial Trading Systems (/articles/applications/confidence-governed-trading)
- Rights-Grade Content Generation With Provenance Tracking (/articles/applications/rights-grade-generation)
- EU AI Act Structural Conformity Through Architecture (/articles/applications/eu-ai-act)
- Autonomous Vehicle Embodiments (/articles/applications/vehicle-embodiments)
- Cross-Domain Application Embodiments (/articles/applications/infrastructure-embodiments)

APPLICATIONS · GENERAL

- Autonomous Vehicle Full-Stack Governance From Sensor to Motor (/articles/applications/autonomous-vehicle-governance)
- Auditable Engagement Authorization for Autonomous Weapon Systems (/articles/applications/defense-engagement-authorization)
- Governed Clinical AI: Closing the Safety and Compliance Gaps in HIPAA, FDA SaMD, and ONC Interoperability (/articles/applications/healthcare-full-stack)
- Governed AI for Financial Services: An Auditable Full-Stack Architecture for Trading, Risk, and Compliance (/articles/applications/financial-services-full-stack)
- Full-Stack Cognition Architecture for Education (/articles/applications/education-full-stack)
- Governed Full-Stack AI Architecture for Smart City Infrastructure (/articles/applications/smart-city-full-stack)
- Governed AI for Smart Manufacturing: Closing the Compliance Gaps in ISA-95, IEC 62443, and Catena-X (/articles/applications/manufacturing-full-stack)
- Audit-Grade AI for Precision Agriculture: Governed Compliance Evidence Across Crops, Livestock, and Autonomous Equipment (/articles/applications/agriculture-full-stack)
- Governed Autonomy Architecture for L4/L5 Autonomous Vehicle Regulatory Compliance (/articles/applications/autonomous-vehicle-domain)
- Smart-Yard and Port Operations: Federating Container Custody Across Siloed Terminal and Yard Systems (/articles/applications/smart-yard-domain)
- Governed Surgical and Clinical Robotics: A Compliance Architecture for AI-Enabled Medical Devices (/articles/applications/medical-robotics-domain)
- Auditable Defense Autonomy: Structural Governance for Autonomous Weapon Systems (/articles/applications/defense-platform-domain)

APPLICATIONS · SPECIFIC

- Waymo vs Governed Cognition: A Full-Stack AV Alternative (/articles/applications/waymo-full-stack)
- Anduril Lattice Alternative: Governed Defense Autonomy Beyond Platform Coordination (/articles/applications/anduril-defense)
- Epic Systems Alternative: Governed Clinical AI Beyond Server-Side Rules (/articles/applications/epic-systems)
- Bloomberg Terminal AI vs Governed Financial Agent Execution (/articles/applications/bloomberg-terminal)
- Tesla Robotaxi vs Governed Autonomous Driving: The Cognitive Architecture Gap (/articles/applications/tesla-robotaxi)
- Lockheed Martin Defense AI vs Governed Engagement Authorization (/articles/applications/lockheed-martin)
- Siemens Healthineers Alternative: Governed Diagnostic AI Beyond In-Workflow Assistance (/articles/applications/siemens-healthineers)
- Palantir AIP vs Governed Operational AI: The Cognitive-Architecture Layer (/articles/applications/palantir-aip)
- C3 AI Alternative: Governed Cross-Domain Coherence Beyond Enterprise AI Applications (/articles/applications/c3-ai)
- UiPath Alternative: Governed RPA Beyond Robotic Execution (/articles/applications/uipath)

Applications overview → (/applications)