AIONdb

AIONdb: Architecture of a Shared Memory Substrate

The Continuity Engine for Coordinated Intelligence

A detailed look at the architecture, engineering decisions, and capabilities behind AIONdb — a new kind of middleware for AI agents in the enterprise, purpose-built for distributed autonomous systems operating in the real world.

Mike Monteith · Founder & Chief Architect · May 2026

The Pattern Across 40,000 Years

Every leap in human coordination — cave walls, tablets, the printing press, bulletin boards, enterprise databases, social networks — follows the same story. Someone builds a better shared surface for knowledge. Not a better message, not a better messenger. A better shared surface, or "blackboard."

Each new blackboard solved a limitation of the previous one and left something out for the next. The cave wall was persistent but not portable. The tablet was portable but not replicable. The printing press was replicable but not

bidirectional. The community bulletin board was bidirectional but structureless. The enterprise database was structured but siloed. The social network was global but had no coherence guarantee.

Two new participants are arriving in this lineage at the same time: physical systems that need to coordinate with each other in real time (hospitals, energy grids, industrial facilities, transit networks, smart buildings, etc.) and AI agents that need to coordinate with each other and with the physical systems they touch. Both need a blackboard with every property every previous blackboard provided — and properties no previous blackboard has ever offered: transactional coherence, temporal ordering, cryptographic auditability, physical safety, and the unity of structured knowledge with semantic understanding.

The Blackboard, Re-Inherited

The word *blackboard* is doing double duty above. It is both the 40,000-year metaphor and a technical term of art. The technical lineage is worth surfacing, because AIONdb is its descendant.

In the 1970s at Carnegie Mellon, the HEARSAY-II speech-understanding system gave its name to **blackboard architecture**: multiple specialised knowledge sources writing to a shared structured memory, reading each other's contributions, with no central controller. The pattern bridged AI's two traditions of the era — the *neats*, who held that intelligence required formal logical structure, and the *scruffies*, who held that it required heuristic competence woven from many small, imperfect sources. Hearsay's blackboard let scruffy components contribute to a neat structure. Tuple spaces (Gelernter, 1985), the Actor Model (Hewitt, 1973), and modern conflict-free replicated data types descend from the same lineage.

The pattern was set aside in the 1990s, not because it failed, but because the orchestration model — central conductor, request/response, service bus — was easier to build with the tools and economics of that decade. The era of LLM agents has reopened the original question. The agents are scruffy. The

hospital, the breaker, the audit trail are neat. The architecture that resolves the tension already exists in the literature ; what the era requires is an industrial-grade implementation of it.

AIONdb is the blackboard re-inherited — rebuilt for the era when the scruffy components are LLM agents and the neat structure must hold up to a hospital audit.

The Human Middleware Crisis

Most of the world's critical data is currently locked in legacy systems and silos. These systems were designed to support humans operating in "human time" — built with screens and reports for a person to look at, interpret, and manually act upon. Because these systems cannot talk to each other in real time, organizations have relied on people to bridge the gaps. We call this **Human Middleware**: highly trained professionals forced to act as the manual integration layer between disconnected tools.

The arrival of AI agents has pushed this structural flaw to a breaking point. We are attempting to unleash machines that operate at millisecond speed on top of legacy infrastructure that requires human pauses. The world's leading analysts have independently confirmed the diagnosis. Bain & Company writes that the gap in agentic AI today "isn't in ambition; it's architecture." Gartner and Deloitte both predict that more than 40% of agentic AI projects will fail by 2027, citing legacy infrastructure incompatibility as the cause. Bessemer identifies the memory and context layer as an emerging infrastructure category in its own right. McKinsey reports that fewer than 10% of enterprises have successfully scaled AI agents to deliver tangible value.

> **Intelligence is not the bottleneck. The infrastructure beneath it is.**

Four Failures of Distributed Autonomous Systems

The same four failures appear wherever autonomous systems are asked to coordinate at scale. Enterprises running AI pilots in 2026 are hitting them. Hospitals running multi-vendor medical systems have been hitting them for decades. Industrial operators, utility companies, and building portfolios encounter them every day. AI agents have not introduced these problems. They have inherited them and made them acute.

MEMORY LOSS ACROSS SESSIONS, VENDORS, AND TIME

In AI agent deployments, the model forgets everything between sessions. Each conversation starts from scratch. Each agent loses its accumulated knowledge when the model restarts, the API times out, or the vendor changes.

In hospitals, every system has its own data model. The patient monitoring system from one vendor cannot share state with the medication administration system from another. The accumulated institutional knowledge of how a specific patient responds to a specific treatment exists in scattered systems and human memory, not in any shared substrate.

The industry is trying to solve the AI version at the model layer with longer context windows, memory features, and persistent threads. The hospital version is being patched with integration middleware. Both approaches are architectural errors. They couple knowledge to a specific vendor's proprietary system. Switch vendors and the memory disappears.

ORCHESTRATOR FRAGILITY

When multiple AI agents coordinate a task, today's frameworks wire them through a central coordinator. When one agent's network goes down, which happens regularly, the entire pipeline crashes.

When a building management system tries to coordinate HVAC, lighting, security, and elevator systems, the integration layer becomes a single point of failure. When it fails, every dependent system fails with it. The same pattern repeats in industrial control systems, energy management platforms, and transit networks.

Cascading timeouts. Fragile coupling. One point of failure. These are the failure modes the distributed-systems community solved decades ago. Both the AI agent ecosystem and the industrial systems integration ecosystem ignored that literature and reinvented the 1990s enterprise service bus.

THE AUDIT GAP

Every regulated industry — healthcare, financial services, energy, transportation, government — lives or dies on the same compliance question: *who knew what, and when?* Today's AI systems answer with log files. A log file is editable, replayable, and forensically weak. When a regulator or a plaintiff's lawyer asks for the chain of custody on an AI-driven decision that affected a patient, a portfolio, a building, or a power grid, the operator faces a choice between proving custody and producing it.

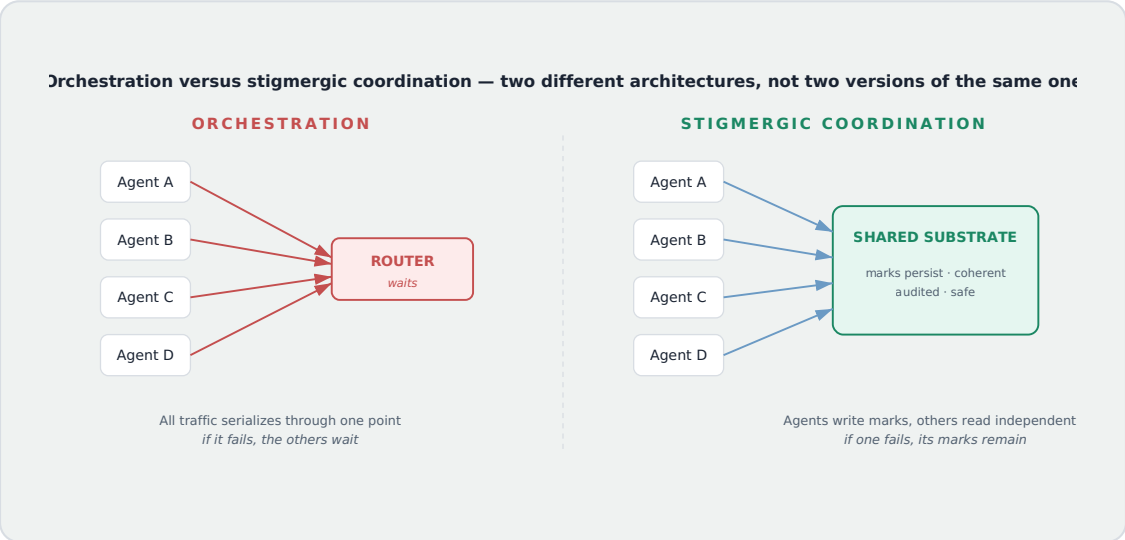
In hospitals, the chain of custody for a treatment decision must withstand a malpractice review years after the fact. In financial services, it must withstand a regulator examining the trade. In government, it must withstand a freedom-of-information request. The difference between an audit log and an audit *proof* is the difference between an organisation's word and a mathematical guarantee. AI agents have not introduced the audit gap. They have widened it past the point where the tools the industry currently has can bridge it.

PHYSICAL LIABILITY

When AI agents are connected to physical systems — hospital equipment, industrial controls, building infrastructure — the question of what happens when an agent gets it wrong moves from theoretical to urgent. Every deployed AI

system's safeguards are software: prompt engineering, constitutional AI, RLHF. Software checking software. These approaches can block a toxic word in a chatbot response, but they are useless at stopping a hallucinating agent from issuing a dangerous command to a high-voltage breaker, an IV pump, or an HVAC actuator.

The same problem exists without AI. Industrial control systems have been built for decades with software-only safety layers. The Therac-25 disaster, where software was the sole safety barrier between computation and physical harm, remains the canonical case study. AI agents do not introduce this risk. They scale it.



The One Radical Choice

AIONdb is built on a single architectural decision that makes everything else possible. Every string entering the system — every name, every relationship, every value — is converted to a 32-bit integer at the boundary and never touched as a string again until output. A triple is twelve bytes. Three integers. The query engine operates on integer comparisons with no memory allocation in the critical path, no garbage collection, and no cache thrashing.

This decision, drawn from 18 years of architecting life-critical distributed systems, is why AIONdb delivers **770,117 results per second** on a 2023 MacBook Pro M2 — a single laptop rather than a cluster. It is why a cold-boot two-node cluster running the full production security path (API-key authentication, TLS 1.3, per-context role-based access control, and encrypted query audit) holds **2,164 queries per second** with a measured 3% throughput tax against the non-security path. It is why the W3C SPARQL 1.1 conformance suite passes **328 of 328 tests**. And it is why GPU acceleration works, because integer arrays are GPU-native with no marshalling required.

Every architectural commitment that follows — the Guardian Cycle, the cryptographic audit ledger, the physical-safety probes, the agent-memory ontology, the zero-copy WebAssembly participant surface — either depends on integer interning or is made incredibly efficient by it.

The Guardian Cycle

AIONdb is a reactive knowledge graph with a three-phase execution model — the **Guardian Cycle** — that enforces the coherence guarantees a shared substrate requires.

1. Tick. New information arrives — a sensor reading, an agent's observation, a human correction — and the system assigns a monotonically increasing sequence number and writes it to a durable write-ahead log.

2. Guardian. The phase that does not exist in any other system. The database becomes invisible to observers. Rules evaluate against the new information — threshold violations, drug interactions, contradictions, consent revocations. Truth is made coherent before anyone sees it.

3. Tock. Every observer is notified simultaneously. Pharmacy agents, insurance validators, LLM participants, and human dashboards all see the same coherent

snapshot at the same sequence number. A slow observer never blocks a fast one.

Act follows the three phases. Act is the observer's response entering the next cycle as a new Tick, closing the loop. Reality flows in, truth is made coherent, coherent truth flows out, and actions become inputs.

On March 12, 2026, something extraordinary was observed. Three frontier models from three competing vendors — Anthropic's Claude, OpenAI's GPT, and Google's Gemini — coordinated through AIONdb on an extended task without an orchestrator. No central conductor. No APIs between them. Each model wrote its observations to the shared substrate. Each read the others' observations independently. Coordination emerged from the environment, not from instructions. Three competing AI systems, built by three different companies, working together through a common blackboard. The architecture worked exactly as designed.

The same architectural pattern has held in production multi-vendor physical systems for years — different vendors, different protocols, different update frequencies, one coherent substrate.

Four Pillars. Four Failures Answered.

AIONdb's architecture delivers four capabilities that directly address the failure modes above. Each capability solves the failure for both physical systems and AI agents because the failure was never specific to either.

CONTINUITY

Memory that survives the model, the vendor, and the year.

Intelligence lives in the substrate, not inside the system that wrote it. A hospital's vibration monitoring system writes an observation into the graph: *"the compressor in Room 302 is showing vibration patterns consistent with bearing failure."* That observation is a triple: durable, queryable, cryptographically signed. When the monitoring vendor is replaced, the knowledge is still there. When the hospital adds an AI agent five years later to do predictive maintenance, the agent reads the substrate and inherits everything the previous systems knew.

Memory is not a model capability or a vendor feature. Memory is infrastructure. **You can swap AI models like lightbulbs; the institutional memory stays in the wiring.**

Answers Memory Loss.

COORDINATION

Substrate coordination, not orchestration.

Consider an industrial facility where energy distribution, equipment maintenance, and safety systems need to coordinate. Today, an integration platform sits in the middle, translating between systems. When it fails, everything fails. With AIONdb, each system writes to the shared substrate and reads from it. The energy distribution system reads the maintenance system's anomaly flags. The maintenance system reads the safety system's lockout state. None of them know about each other directly. They share an environment.

The same pattern works for AI agents managing a hospital discharge: pharmacy review, insurance validation, transport scheduling. Each watches the graph for patterns it cares about and writes its own contribution back. When one fails, the others keep working. When it recovers, it reads the write-ahead log to catch up.

Zero lines of integration code. Zero points of failure. The substrate is the coordinator.

Answers Orchestrator Fragility.

AUDIT

Cryptographic proof, not log files.

Every mutation, every fact written, every decision recorded, every command issued is cryptographically signed and recorded in an immutable audit ledger. This isn't a log file that can be edited. It's a chain of cryptographic hashes (Ed25519-signed Merkle blocks) where modifying a single byte breaks the entire chain detectably. Independent auditors can verify any entry directly through the substrate's audit verification path.

The audit subscriber lives in the same address space as the query engine. The cryptographic proof costs no extra service, no extra network hop, no extra deployment. Legal-grade audit at substrate cost. For regulated deployments — healthcare, financial services, government — this is not a compliance add-on. It is the transaction path.

When a regulator asks "who knew what, and when?" the answer is a mathematical proof, not a log file.

Answers the Audit Gap.

SAFETY

Deterministic physical containment.

Before any command — AI-issued or otherwise — reaches a physical device, AIONdb's safety probes verify that the device is in a safe state to receive it. For a Modbus-controlled breaker, that means a `Read Holding Registers` command at function code `0x03`. The probe confirms the device's state. If the breaker is in maintenance mode, with a contractor on the line, the command is rejected at the hardware boundary. The audit ledger records a cryptographically signed violation.

This applies whether the command originates from an AI agent, an automated control system, or a human operator. The safety layer does not care about the source. It cares about the state of the physical device.

This is not a software guardrail. This is deterministic physical containment.

The same engineering principle that has governed safety in high-consequence physical systems for decades: the safety mechanism must be independent of the system it is protecting. AIONdb's safety architecture is informed by 18 years of building life-critical distributed systems, where these failure modes are not theoretical. It is also why these systems can be priced by an actuary in a way that probabilistic systems cannot: a deterministic safety probe has a quantifiable failure mode. An LLM's judgment, or a software-only industrial control loop, does not.

Answers Physical Liability.

A New Kind of Middleware

To solve these failures, the market has converged on orchestration: top-down spiderwebs of API calls trying to force disconnected agents to talk to each other. These are fragile by design. Single points of failure, brittle integrations, and no shared truth between agents. When the orchestrator breaks, the system stops.

> AIONdb is a new kind of middleware for AI agents in the enterprise. Unlike traditional middleware, AIONdb is an intelligent **underlay** for AI agents and an **overlay** for existing systems and data.
>
> While others are building brittle strings between agents, AIONdb is the concrete floor they all stand on.

The substrate sits beneath the agents and on top of the systems they need to work with. As an *underlay*, it gives agents the shared memory, coordination, and safety they don't get from each other. As an *overlay*, it pulls existing systems into a unified graph through built-in connectors for each protocol — BACnet for buildings, HL7 and FHIR for hospitals, Modbus for industrial controls, MQTT and SNMP for telemetry, MCP for AI agents, and the rest. Agents don't connect to each other. They connect to the substrate. They write what they know, read what they need, and discover the truth on a shared blackboard. There is no central conductor. If one agent fails, the others continue working through the same shared environment. New agents join by reading the substrate, not by integrating with every other agent in the system. **AIONdb is the policy enforcement layer between agents and the systems they need to touch.**

The systems agents need to coordinate with were not built with agents in mind. *Legacy IT* in this conversation is two things, not one. On the business side it is the operational systems of record — Epic and Cerner in hospitals, SAP and Oracle in industry, Salesforce and ServiceNow across the enterprise — built around screens, forms, and workflows that assume a human is reading, deciding, and acting. On the physical side it is the operational technology —

BACnet building controls, nurse-call systems, vital-signs monitors, Modbus-controlled industrial equipment, fire-and-access systems, SCADA — built around field-bus protocols, polling cycles, and operator consoles that assume a human is supervising. Both were designed for human pace, human consent points, and human accountability. Agents arrive at millisecond speed and overwhelm every one of those assumptions.

The deeper problem is trust. Even when an agent can technically reach a legacy system, no security architect is going to authorize unrestricted write-back. The risk is twofold. On the business side, an agent writing into an EHR, a billing system, or a CRM without policy enforcement at triple-level granularity is a regulatory and audit catastrophe waiting to happen. On the physical side, an agent writing a setpoint into a BACnet HVAC controller or a Modbus breaker without a deterministic safety check repeats the Therac-25 architectural error at machine speed. AIONdb resolves both at the same boundary. Agents write to the substrate, not to the underlying systems. The substrate enforces who can write what, where, and under what conditions, at per-context role-based access granularity. Every mutation is cryptographically audited. Every command that would actuate a physical device is verified at the protocol level before reaching the wire. **A smart, deterministic, dynamic enterprise cache that does policy-based enforcement of agent integration with audit and safety** — exactly what the security architect has been looking for, exactly where they expected to be told it doesn't exist.

Existing systems — hospital information systems, building management platforms, nurse-call and vital-signs networks, industrial control networks, enterprise databases — remain where they are. AIONdb does not replace them. Your existing infrastructure stays. AIONdb makes it usable for AI. **And here the underlay/overlay duality resolves:** the same substrate that overlays your legacy landscape becomes the surface your agents read from. The overlay becomes the underlay. The two markets named below — digital-twin infrastructure for the physical world and shared memory for the AI agent world — are not adjacent products ; they are the same architecture viewed from opposite directions.

Two Entry Points, One Architecture

The same substrate addresses two markets that arrive from opposite directions and discover they need the same thing.

THE PHYSICAL WORLD

Digital-twin infrastructure

Hospitals, buildings, factories, energy grids, and transit networks need a single, coherent, temporally ordered, cryptographically auditable reflection of physical reality, shared by every sensor, every operator, and every system reasoning on top of it. Not a federation of disconnected dashboards. AIONdb turns 18 industrial, healthcare, enterprise, and AI protocols (BACnet, Modbus, MQTT, HL7, FHIR, SNMP, MCP, and more) into one truth surface. Configuration time, not development time. The physical world reported faithfully, continuously, with an audit trail legal counsel can accept.

This is the market with the longer history, the larger established budgets, and the most concrete operational pain.

THE AI AGENT WORLD

Shared memory for autonomous systems

Autonomous agent systems need more than retrieval. They need continuity across sessions, coordination without orchestrators, audit trails that hold up to regulatory scrutiny, and safety for systems that touch the physical world. What was observed. What was decided. Who has authority. What must never reach the wire without physical verification. AIONdb provides this as infrastructure rather than as a feature of any single model or framework, which means the memory persists regardless of which models are running, which vendors are involved, or how the agent ecosystem evolves.

This is the market the leading analysts and consultancies have spent the past year diagnosing. Bain, Gartner, McKinsey, Deloitte, and Bessemer have all identified the same gap.

The convergence is the proof. Both entry points lead to the same architecture, which is the strongest demonstration that the substrate is general-purpose rather than a point solution. A substrate coherent enough to model a hospital is coherent enough to be the memory layer for the agents that work inside it. A substrate that handles real-time physical actuation safely is a substrate that handles AI-driven actuation safely. The agent memory capability is not a separate product. It is a derived property of an architecture that already works in the real world.

The Evidence

Extraordinary claims need evidence. The slate below is the cold-boot measurement from 2026-05-05 — all numbers reproducible from timestamped reports in the codebase.

350B	328/328	795K	2,235
ERROR-FREE RESULT TRIPLES ZERO FAILURES	W3C SPARQL 1.1 COLD-BOOT VERIFIED	RESULTS PER SECOND SINGLE-NODE ENGINE	QPS FULL SECURITY TWO-NODE CLUSTER

Single-node engine. 794,908 results per second on the 68-query institutional stress suite, measured at cold boot on 2026-05-05 — a +3.2% lift over the prior 770,117 r/s record from 2026-04-20.

Two-node HA cluster, non-security path. 2,315.2 queries per second on the 100,000-query stress run, 25 threads, binary wire format. Mean client latency 10.55 ms.

Two-node HA cluster, full production security path. 2,235.4 queries per second with API-key authentication, TLS 1.3, per-context role-based access control, and AES-256-GCM encrypted query audit all active simultaneously.

Mean client latency 10.91 ms. The security tax versus the non-security path is **−3.45% throughput and +0.36 ms latency** — measured inside the PHIPA / Threat-and-Risk-Assessment regulatory envelope.

Split-hardware one-million-query endurance. 2,371 queries per second sustained (M2 server, Mac Mini M1 harness, 40 Gb Thunderbolt link), locating the 12-core M2 hardware ceiling at approximately 2,354 qps. The engine is CPU-bound against the die, not against the harness.

Correctness. 328/328 W3C SPARQL 1.1 conformance. 68/68 institutional stress suite. 4,465 workspace tests passing with zero failing.

> **200 million queries. 350 billion error-free result triples. Zero failures.**

Across 1,700+ timestamped report files spanning four months — 100,000-query stress runs, ten-million-query endurance runs, fifty-million-query marathons, deliberate leader-kill chaos injections every seven seconds, full production security paths with authentication and TLS and encrypted audit all active — the system has not produced a single failed result. Three numbers, each separately auditable: the query-invocation count against the reports directory, the result-triple count against the sum of per-run *Total Results* banners, and the failure count against zero.

Codebase. 193,527 lines of Rust. Not Python glue around someone else's library. Rust — the language you choose when failure is not an option.

Intellectual property. Provisional application USPTO 64/006,261, filed March 15, 2026, 102 claims across eight invention families. Utility filings are documented under separate cover.

What Comes Next

For forty thousand years, we have been building better blackboards, each one more persistent, more portable, more scalable, more structured than the last. Each transition preserved the pattern and added a property. Each transition also left something out, and the thing that was left out became the defining problem of the next era.

Two new participants are joining this lineage at the same time. Physical systems that need to coordinate continuously, safely, and across vendors. AI agents that need to coordinate without orchestrators, remember across sessions, leave a verifiable audit trail, and be trusted to touch the real world. Both need a substrate with transactional coherence, temporal ordering, cryptographic auditability, physical safety, and the unity of structured knowledge with semantic understanding. Neither has had one.

AIONdb is that substrate. The architecture draws on 18 years of building life-critical distributed systems — the environments where these failure modes are not theoretical. The substrate itself has been validated through 350 billion error-free result triples and a documented multi-vendor agent coordination demonstration. The world's leading analysts and consultancies have spent the past year describing a gap. AIONdb is built to close it. The same architecture, proven against both worlds, ready for the production deployments they require.

THE FOUNDER

AIONdb was created by Mike Monteith, a Canadian technology builder, inventor, and systems architect. For 35 years, Mike has been focused on solving how distributed, autonomous systems coordinate truth. Mike founded and led ThoughtWire, a Canadian company focused on digital twin technology, for over 15 years, building real-time operating platforms for life-critical infrastructure at scale — smart hospitals, large commercial environments, and complex physical systems.

AIONdb is the architecture Mike has been converging toward across two companies and three codebases.

Mike Monteith — Founder & Chief Architect

Go Deeper

Speak with the founder →

Direct access to the founder and chief architect. No sales funnel.

Read the white paper →

The full argument for why AI coordination requires a shared substrate.
