Unified Agentic Memory Layer for AI Coding Agents

Enterprise AI Development Infrastructure & Research-Grade Agent Framework

Version 2.0 | January 2026

Executive Summary

Modern AI coding agents—including Cursor, Claude Code, Codex, Kiro, and emerging IDE/CLI/cloud-based platforms—demonstrate remarkable individual capabilities but face a critical architectural limitation: they operate in isolation without persistent, shared knowledge of project context, architectural decisions, or coordination state across sessions and parallel workstreams.

This paper presents a **vendor-agnostic unified agentic memory layer** designed to provide persistent code intelligence, structured decision logs, and interoperable connectivity. By implementing this layer, any coding agent can immediately leverage the accumulated knowledge and decisions of previous agent executions while maintaining enterprise-grade auditability and governance.

The proposed architecture is intentionally protocol-centric, treating agents as interchangeable clients and standardizing on open interoperability primitives—including Agent2Agent (A2A), Model Context Protocol (MCP), and the LangChain Agent Protocol—to maximize ecosystem compatibility and facilitate enterprise adoption.

1. Problem Statement

AI coding assistants remain fragmented and frequently stateless across project lifecycles. This fragmentation creates several critical challenges:

- **Knowledge Silos**: Without a shared memory substrate, parallel or sequential agents duplicate work, introduce contradictory changes, and lose the rationale behind architectural decisions.
- **Coordination Failures**: Agents cannot safely coordinate parallel efforts without explicit conflict detection and resolution mechanisms.
- **Governance Gaps**: Enterprises lack consistent controls over agent-driven changes, including attribution, audit trails, and policy enforcement.
- **Context Loss**: Each agent session begins without awareness of previous decisions, rejected alternatives, or documented trade-offs.

1.1 Design Goals

G1. Universal Agent Integration

Any agent—whether IDE-embedded, CLI-based, or cloud-deployed—can read project context and write decisions and changes through standardized interfaces.

G2. Hybrid Retrieval Semantics

Support semantic recall via vector similarity, structured reasoning through knowledge graphs, and temporal queries with complete audit trails, enabling agents to understand "what was done," "why it was done," and "when it was done."

G3. Safe Multi-Agent Coordination

Provide primitives for mutual exclusion, leases, task claims, and conflict detection to enable parallel agent execution without race conditions or architectural drift.

G4. Enterprise-Grade Security & Governance

Implement authentication (authn), authorization (authz), multi-tenancy,

encryption, and policy enforcement to meet enterprise compliance requirements and preserve auditability.

G5. Research-Grade Observability

Enable reproducible agent runs, reasoning summaries, dataset export, and evaluation hooks to support continuous improvement and systematic evaluation of agent behavior.