SIGNAL 1

AGENT CONTROL PLANE

Reference Architecture for Enterprise Agentic AI in Healthcare

Version 1.0 — July 2026. This is a living document. The high-level reference architecture is designed to stay stable, but the industry is evolving fast — the component must-have functions, and the available options, will keep changing — so every revision of this detailed technical design is recorded in the changelog with the reasoning behind the change.

SIGNAL 1

Table of Contents

1 The problem this architecture solves

An AI agent is software that acts: it reads from your systems of record, reasons over what it finds, and writes back into the systems: schedules, orders, messages, claims.

Your health system will not run one; it will run a fleet, built by your own teams, embedded in your enterprise software, and delivered by your vendors. Run that fleet without a management layer and the failure modes are predictable, because they are already visible today: agents in production that nobody has inventoried; governance committees approving what they cannot see and rubber-stamping what they cannot measure; an auditor's question that nobody can answer; incidents that cannot be attributed to a specific agent, version, and policy; and agents that repeat the same mistake for months because clinician corrections have not been fed back to the agent.

The reference architecture prevents these by construction. Its organizing idea is a loop: **collect** everything every agent does, **evaluate** it against your own expectations, policies and workflows, and use what you learn to **improve** the agents. Every component described below exists either to run one beat of that loop or to make it trustworthy.

2 Why healthcare needs its own reference architecture

The consequences, systems, and standards governing agent behavior in healthcare are materially different from those in a typical enterprise. Healthcare is a dynamic, high-stakes environment in which the correctness of an agent's action depends on the current patient, workflow, policy, role, and operational situation.

A response that is appropriate in one context may be unsafe in another, so agents cannot be evaluated only for latency, cost, toxicity, or the quality of an isolated output. They must be assessed against the institution's current clinical and operational context: whether they followed the right SOP, used the appropriate tools, acted in the correct sequence, recognized when conditions had changed, and involved the right clinician or operation staff when required.

Governance must remain connected to that operational reality, with each agent's approvals, risk tier, access, evaluations, incidents, and human interventions continuously reflected in its governance record rather than reviewed only at deployment. Failures can create substantial financial and operational costs and, most importantly, harm patients. Healthcare-specific evaluation must therefore do more than identify mistakes after the fact; it must turn incidents, clinician corrections, and near misses into actionable improvements that change future behavior, so the same failure is not repeated.

Auditability, continuous governance, human oversight, clinical-system integration, whole-trajectory semantic evaluation, and continual learning must therefore be built into the architecture from the start.

3 How to read the diagram

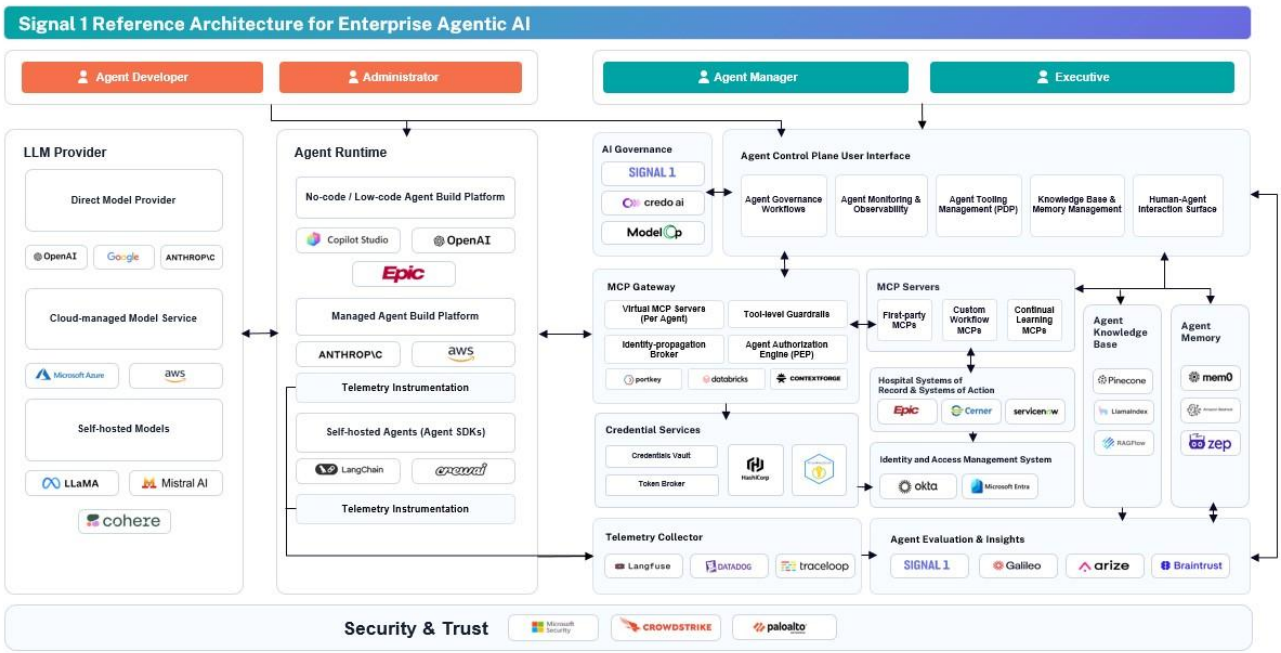

Figure 1. The Agent Control Plane reference architecture — v1.0, July 2026.

The architecture depicts how the full enterprise stack works together across an agent's lifecycle, from creation through operation, monitoring, governance, and improvement.

Agent developers build agents in no-code platforms, managed agent build platform, or self-hosted agent frameworks, using models supplied directly, through cloud-managed services, or from self-hosted infrastructure. Regardless of where an agent runs, it is instrumented to emit telemetry and routes its actions through the control plane. The MCP Gateway is the enforcement point: it identifies the agent and the user or service on whose behalf it is acting, applies tool-level guardrails and authorization policy, obtains short-lived credentials through credential and identity services, and brokers access to approved MCP servers and hospital systems of record and action.

The control plane should integrate with, rather than replace, the hospital's existing technology stack. It should rely on the organization's IAM platform for workforce identity and group membership, its credential vaults and token services for secrets and delegated access, its cybersecurity tools for threat detection and incident response, and its existing EHR, ERP, CRM, scheduling, claims, data, and operational systems as the authoritative systems through which agents read and act. The user interface above these services gives agent managers and executives a common place to govern the portfolio, monitor behavior, manage tools, curate organizational knowledge and memory, and intervene when human judgment is required. Every trace, tool call, decision, and outcome is collected into the telemetry layer and evaluated against the health system's policies, workflows, and performance expectations, producing insights that feed back into governance, agent configuration, shared knowledge, and continual improvement. Security, identity, audit, and enterprise trust controls therefore span the entire architecture, while the control plane provides the common management layer that connects and governs the technologies the hospital already owns.

What follows walks the architecture in the order it would be used: stand up the control plane once, onboard each agent through it, then operate the loop continuously.

4 Phase 1 — Stand up the control plane

These components go in once, before the first governed agent, and every agent you deploy afterwards inherits them automatically.

The MCP Gateway is the vendor-neutral entry point between every agent and every hospital system. Nothing an agent does to a system of record should happen except through it. Four capabilities matter. **Virtual MCP Servers (Per Agent)** give each agent its own private endpoint exposing exactly the tools it has been granted, least privilege by construction, not by convention. The **Identity-propagation Broker** binds every action to a specific agent identity and, where the agent acts for a person, to the on-behalf-of human. Token exchange is leveraged rather than a shared service account, so accountability survives into the downstream system's own logs. **Tool-level Guardrails** enforce deterministic, pre/post-execution rules: hard stops the agent cannot talk its way past, because they are predicates on the call rather than a second model's opinion. The MCP Gateway give you a central kill switch, revoking a tool or an agent in one place, effective immediately. The **Agent Authorization Engine** is the policy enforcement point: access policy is decided in the management interface and enforced here, in line, on every call. Deploy the gateway in your own tenancy: it concentrates credentials to your source systems, and that concentration should never sit outside your boundary. Much of the integration reality it fronts (HL7 v2 interfaces among them) was never designed to traverse the public internet.

Credential Services. The gateway's companion is a **Credentials Vault** and **Token Broker**: credentials to downstream systems live encrypted in the vault, and the broker issues short-lived, scoped tokens per agent, per tool. All of it syncs with the **Identity and Access Management System** you already run. The control plane extends your identity stack to agents as a new class of actor; it does not replace it.

The Telemetry Collector goes in with the gateway so that trace capture starts working as soon as we connect agents to the gateway. Instrumentation is OpenTelemetry-based: runtimes you control emit full traces through an SDK, and closed runtimes are still covered by the gateway, which records every tool call.

The **Agent Control Plane User Interface** is where people oversee and control the agent ecosystem. At its foundation is the **Agent Registry**, the authoritative inventory of agents observed across the enterprise and updated automatically from gateway traffic and OpenTelemetry traces. Each agent record captures key information such as ownership, risk tier, evaluation performance, cost, and approved tool-access scope. From this central interface, teams can execute and track **governance workflows**, including reviewing and approving new agent requests; **monitor agents' behavior and performance** and investigate issues; **manage the tools and permissions** that allow agents to read from and write to hospital systems; and curate the **knowledge base and memory layer** that ground agent behavior in organizational context. The interface also includes the **human-agent interaction surface**, where clinicians and operational staff can work directly with agents, review human-in-the-loop requests, approve or reject proposed actions, and provide corrections that feed back into evaluation and continuous improvement.

The Agent Control Plane is where **AI governance** becomes operational. Governance teams define the policies, approval requirements, human-review rules, and access constraints that should apply to each class of agent; the control plane translates those decisions into controls enforced by the gateway and continuously verifies that agents remain within them. Operational evidence then flows back into the same governance process through evaluation results, incidents, overrides, tool usage, cost, and audit history.

Security and Trust remain the enterprise foundation beneath the control plane. Runtime protection, secrets and data protection, network security, threat detection, and incident response must extend to agents as a new class of actor, including AI-specific risks such as prompt injection and malicious tool use. The control plane integrates with these existing capabilities and strengthens them by supplying scoped identities, authorization and guardrail decisions, detailed telemetry, and a complete audit trail.

5 Phase 2 — Onboard an agent

With the plane standing, onboarding becomes a repeatable process rather than a project.

Register the MCP servers the agent will need. Tools reach agents as MCP servers, in three patterns: **First-party MCPs** your system vendors ship; **Custom Workflow MCPs** your teams build; least-privilege tools that wrap a clinical or back-office workflow rather than exposing raw APIs; and other servers that improve agents' performance, most notably **Continual Learning MCP tools**. Registration never replaces a vendor's native server: the gateway discovers the server's tool schemas and re-exposes them wrapped — the same tools, now cataloged, allowlisted, and observable. Each tool carries its own risk tier, access scope, and review policy, so builders compose agents from safe actions instead of raw system access.

Request, approve, provision. A team wanting to deploy an agent declares what it is for: the job, the tools it needs, the guardrails, the rollout plan, the success metrics. That intake *is* the governance configuration — the agent specification becomes the compliance specification, which is what lets domain experts publish governed agents without hand-rolling risk-framework, PHI-boundary, and review-policy decisions. Access policy is decided in **Agent Tooling Management**, and on approval the gateway mints the agent's virtual MCP server: its own URL, a rotatable token, credentials scoped from the vault. Nothing is granted broadly; everything granted is recorded.

Ground the agent in the Agent Knowledge Base. Agents must act on current organizational truth: SOPs, policies, escalation paths, role definitions, and on-call rules. These documents must be versioned, and represented in agent-readable form. Versioning is non-negotiable: when an action is questioned later, you need to know which SOP version was in force at the time the agent acted. The same store is read from the other side of the loop; the evaluation framework judges agents against the same policies that guide them.

6 Phase 3 — Operate and improve

This is the loop the architecture exists to run: collect → evaluate → improve, continuously.

Agents governed in real time. Every tool call traverses the gateway: deterministic guardrails gate before execution, and risky or policy-violating actions route to human review *before* they execute. Reviews reach clinicians where they already work, such as EHR messaging, Teams, and existing worklists. Additionally, a dedicated **Human-Agent Interaction Surface** provides a medium for high-bandwidth interaction, carrying sufficient context for decisions to be made meaningfully, and avoid rubber-stamping.

The Telemetry Collector assembles full workflow traces; every step, tool call, and handoff. Traces also carry cost, attributable per call to agent, tool, model, workflow, and department: the basis for spend control and for connecting cost to the tasks that the agent completed.

Agents must be monitored both operationally and semantically. Operational monitoring includes tracking metrics like latency, error rates, and token spend. It tells you *how* an agent executed, and mature horizontal tools do it well. The question that matters in a hospital is different: *did the agent do the right thing?* Agent failures emerge across trajectories, not in single responses, so answering it requires evaluation of aggregated traces against your own context (i.e. the SOPs, escalation rules, and expected outcomes in your knowledge base) applied to the agent's entire trajectory. In practice, that means evaluation metrics and rubrics must be generated per agent based on its declared job and its grounding documents; continuous scoring of task completion, tool-use correctness, and policy alignment. A risk-tiered cadence can be implemented, with higher-risk agents evaluated more often. For every LLM-as-Judge determination, it is crucial to not just have a bare score, but also a rationale you can drill into. The evaluation framework should also support regression testing whenever a model, prompt, tool, or SOP changes, so behavior does not silently break after an upgrade. Because evaluations, traces, approvals, and policy versions all accumulate against each agent's registry record, audit-ready compliance evidence becomes a by-product of operations rather than a quarterly scramble.

Extract insights; create actionable memories. Evaluation findings and clinician corrections are distilled into agent memory: scoped to patient, department, or hospital; access-controlled, because memory is data; and periodically consolidated so it stays dense and current instead of accreting noise.

The loop closes through **Continual Learning MCPs**. Memories are served back to agents as a tool, through the same gateway. Any runtime that can add an MCP server, today it's effectively all of them, can consume the memories. So, a correction made once reaches every agent it applies to, whatever platform the agent was built on. This is the property that separates an agent fleet from the last generation of AI deployments: a documented mistake becomes a durable correction, and the fleet measurably improves in production.

7 Choosing your stack

Healthcare adds selection criteria that generic enterprise-software evaluations often miss.

Before building a shortlist, assess every component against the HIPAA Security Rule's core expectations: protect the confidentiality, integrity, and availability of ePHI; control and verify access; preserve auditability; and maintain appropriate safeguards during normal operations and failures.

- **Deployable within your security boundary.** Traces, prompts, credentials, and tool outputs may contain ePHI or provide access to source systems. Prefer components you can self-host, where operationally practical, or deploy within your own cloud tenancy. Require a BAA and clearly defined security responsibilities for any service operating outside that boundary.
- **Traceable and auditable.** Every consequential action, including an allowed tool call, blocked request, human override, or evaluation verdict, should be able to be reconstructed with the identity, context, and policy version in force at the time.
- **Customizable to your institution.** Hospitals differ in policies, role hierarchies, escalation paths, terminology, and risk tolerance. Components must represent those differences without forcing workflows into rigid vendor defaults.
- **Identity and access done properly.** Require distinct identities for each agent, least-privilege authorization, delegated user context where applicable, short-lived credentials, and revocable access. Reject architectures that rely on shared service accounts.
- **Built on open standards.** Favor MCP for tool interoperability and OpenTelemetry for traces. Open standards reduce lock-in and make it easier to replace components as the market consolidates.
- **Operationally mature.** Anything operating in the path of a clinical workflow should be treated as a tier-one application, with appropriate availability, failover, recovery, monitoring, incident response, and service-level commitments.

Component	What to require	Some of the options on the market (July 2026)	Our current read for healthcare
LLM Provider & Agent Runtime	Model- and runtime-neutrality above them; runtimes must support MCP and, where possible, SDK-level trace instrumentation	Models — direct: OpenAI, Anthropic, Google; cloud-managed: Microsoft Azure, AWS; self-hosted: Meta Llama, Mistral, Cohere. Runtimes — Microsoft Copilot Studio, OpenAI Agent Builder, Anthropic Claude Managed Agents, AWS Bedrock AgentCore, LangGraph, CrewAI	Expect to support more than one agent platform, but avoid unnecessary sprawl. Most health systems should start with the platforms already embedded in their environment, such as Epic and their primary cloud provider, then optionally add a frontline model provider such as OpenAI where it offers a clear advantage. Maintain a BAA with each, and ensure the control plane governs them consistently.
MCP Gateway	Per-agent virtual servers; deterministic tool-level guardrails; identity propagation via token exchange; complete call-level audit log	Portkey (now Palo Alto Networks), Databricks, AWS AgentCore Gateway, IBM ContextForge (open source), Rubrik AI Gateway	Prioritize enforceable controls: the gateway should identify the calling agent, restrict which tools it can use, propagate user context where required, and record every call. Because it brokers access to sensitive systems, require clear security boundaries, strong operational controls, and a BAA where applicable. This is the wrong place to accept a black box — it holds your credentials and writes your audit trail.
MCP Servers (the tool layer)	Least-privilege tools carrying risk tier, access scope, and review policy; native vendor servers wrapped, never replaced	First-party servers your system vendors ship as they arrive; custom workflow servers your teams build	Curate, don't accumulate: a governed library of purpose-built workflow tools beats raw API exposure, and tool authoring should itself be a governed activity

Choosing your stack

Component	What to require	Some of the options on the market (July 2026)	Our current read for healthcare
Credential Services & IAM	Encrypted vault; token broker issuing short-lived scoped credentials; syncs with the IAM you already run	IAM: Okta (incl. Okta for AI Agents), Microsoft Entra (incl. Entra Agent ID). Vaults: HashiCorp Vault, Azure Key Vault, CyberArk	Agents require their own identity layer, not only reuse of human identities. The control plane may create and manage an agent identity that is registered with the runtime, while enterprise IAM remains authoritative for users, groups, and organizational access policy. The important requirement is a trustworthy binding among the agent, its owner, its approved purpose, its credentials, and any user on whose behalf it acts. Avoid shared identities and unmanaged service accounts.
Telemetry Collector	OpenTelemetry-native; full-trajectory traces including tool calls and reasoning; per-call cost attribution	Langfuse (open source), Datadog, Traceloop, Arize Phoenix, LangSmith; traces can also land in your lakehouse (Databricks, Snowflake, Microsoft Fabric)	Focus on control over sensitive telemetry: where traces are processed, who can access them, how PHI is minimized or redacted, how long they are retained, and whether they can be exported into the hospital's analytics and security environment. SaaS is viable when the vendor provides the necessary controls, transparency, reliability, and contractual protections.
Agent Knowledge Base	Versioned SOP/policy store with ownership and diffing; agent-readable; the same source evaluation reads	Glean, Shelf, Pinecone, LlamaIndex, RAGFlow, or your existing lakehouse (Databricks, Snowflake, Microsoft Fabric)	Versioning and ownership matter more than retrieval sophistication. Building on the lakehouse you already govern often beats adding a new store
Agent Memory	Provider-agnostic (survives model changes); patient-, department-, and hospital-level scoping; role-based access on reads and writes; periodic consolidation	Mem0, Zep, Amazon Bedrock AgentCore Memory, Anthropic agent memory, LlamaIndex	Memory is where institutional context, user preferences, and prior corrections can accumulate, so provenance and access control are critical. Avoid binding durable organizational memory too tightly to a single model or runtime, and require clear controls over what may be written, retrieved, retained, corrected, or deleted. Shared memory across agents will be important to more efficiently improve your fleet of agents.
Agent Evaluation & Insights	Auto-generated per-agent evaluations; whole-trajectory semantic evaluation against your own policies; risk-tiered cadence; drill-down rationale; regression testing	Signal 1, Arize, Galileo, Braintrust, Patronus AI, LangSmith, Datadog	Support semantic evaluation; assess the agent's full trajectory, not only individual model outputs. It should break the trace into meaningful workflow steps and judge each against the hospital's current policies, SOPs, role expectations, patient context, and escalation rules. The goal is to determine whether the agent completed the task correctly, in the right order, with the right tools, and with appropriate human involvement.
AI Governance	Agent-level (not just model-level) risk records; evidence flowing automatically from operations	Signal 1, Credo AI, ModelOp, Onboard AI, ServiceNow AI Control Tower	Governance should operate as a continuous system, not a one-time approval exercise. Each agent's risk record should remain connected to how it is actually used, including changes in tools, data access, policies, evaluations, incidents, and human interventions, so reviewers can see whether its controls remain appropriate over time.
Security & Trust	Coverage of agents as a new actor class, including prompt injection; consumes the control plane's identities and audit trail	CrowdStrike, Microsoft Defender & Purview, Palo Alto Networks Prisma AIRS, Wiz, Zscaler, your SIEM	Extend the security estate you already run to agents. The control plane rests on it and feeds it.

You can assemble these capabilities from multiple vendors or adopt them as an integrated platform. Either way, the test of a working Agent Control Plane is the one this document opened with: governance, identity, monitoring, and evaluation are established once. Then every new agent, wherever it was built, inherits them automatically.

8 Changelog

v1.0 — July 2026. Initial publication, as the technical companion to the reference-architecture diagram and the Reference Architecture for Enterprise AI in Healthcare paper (both v1.0, July 2026).

Future entries will record what changed and why — a component redrawn, a vendor acquired, a product renamed, a framework release that shifted our guidance.