INTEGRAL CORPORATION TECHNICAL WHITE PAPER

The Architecture Behind Integral Executive AI Teams

How Bounded Agent Teams, Event-Driven Coordination, and Persistent Organizational Memory Turn AI Tools Into a Working Executive Organization

Brett Thomas · Founder, Integral Corporation

August 2026

Executive Summary.

Most organizations still approach artificial intelligence as a collection of disconnected tools: a chatbot for writing, a research assistant, an automation platform, a separate analytics product, and a few experimental agents. This creates the same problem that software sprawl created in earlier technology cycles. Intelligence becomes fragmented, context is lost between tools, and the human leader is forced to become the integration layer.

Integral Intelligence Architecture takes a different approach. It treats AI as an organizational capability rather than a set of tools. Instead of asking a founder to manage a growing collection of agents, the architecture creates a coordinated AI team. A primary Orchestration Agent acts as team leader and chief of staff. Specialist Agents perform defined jobs. All agents remain accessible to the human, but the human primarily works through the orchestration layer. A Shared Brain supplies company context and persistent memory. Skills encode how recurring work is performed. Evals define what acceptable performance looks like. Loops allow work to be checked, repaired, repeated, scheduled, and improved over time.

This edition is written for joint venture partners and technical evaluators. It documents three architectural pillars a technical reader should look for: bounded-context team applications, an event-driven coordination substrate, and a persistent organizational memory layer. Together these three pillars turn a set of agents into a team of teams that a human executive can actually run.

The commercial model matters as much as the technical model. Integral's preferred approach is to build the production system in the client's own environment. The client owns the deployed application, its configuration, data, prompts, skill definitions, workflow logic, organizational memory, and business-specific intelligence. Integral architects, implements, and optimizes the system during the engagement, but the end state is designed for transfer. This turns AI consulting into an asset-building engagement rather than a rented-hours relationship.

I. The Design Problem

SMB leaders increasingly understand that AI can automate work, improve analysis, and increase leverage. The harder problem is organizational. How should multiple agents work together? How should they share context? How should humans supervise them? How can the resulting intelligence remain inside the company rather than disappearing when a consultant, agency, or key employee leaves?

Integral's Agentic Architecture Method defines agentic software as a workflow engine with memory, steps, actions, checks, permission gates, and artifacts. It requires explicit goals, plans, actions, memory, loops, and guardrails. It distinguishes an orchestrator from specialist capability agents and a QA and evaluation role. The

team architecture in this paper extends that model from a single agentic feature to an organizational system of coordinated AI workers, and then further to a team-of-teams model where multiple such systems coordinate through shared substrates.

The architectural objective is therefore not more agents. It is a controlled operating environment in which human leaders can delegate work to a coordinated AI organization with clear roles, persistent context, measurable standards, and safe escalation paths.

II. Design Principles

Five principles constrain every downstream architectural choice. They are stated here so a technical reader can hold the rest of the document against them.

Human at the Helm

Agents propose, humans decide. Consequential actions require explicit approval. The system is designed to increase a human executive's leverage, not to replace judgment. Autonomy is granted along defined authority gradients rather than as a default posture.

Team-Shaped Agent Groupings

Organizational structure maps to agent structure. Agents are grouped into teams that mirror executive functions: Revenue, Operations, Leadership, and AI Executive. Each team has an orchestration agent that plays the role of team lead. This mapping makes the system legible to a founder and gives each team a bounded scope of responsibility.

Upstream-First Sequencing

Revenue capabilities are deployed first so that returns from early phases fund the later phases. This is a sequencing discipline as much as a technical one, but it shapes the architecture. The Revenue team is built to stand alone and produce measurable outcomes before the other teams are added.

Bounded Contexts With Explicit Contracts

Teams do not reach into each other's internals. Each team application owns its own state, specialists, UI, and operational memory. Coordination between teams happens only through two shared substrates: an event bus for verbs (things that happened) and a shared brain for nouns (what is true). This is the same discipline that keeps large service-oriented systems maintainable, applied to a team-of-teams AI architecture.

Client Ownership as End State

Transferability is a design requirement, not an afterthought. Where practical, the production application lives in the client's workspace. The client controls billing, credentials, and infrastructure. The system is documented and evaluated to a standard that allows the client to continue operating it after the engagement ends.

III. The Team Model: One Leader, Many Specialists

Each AI team has one Orchestration Agent and two or more Specialist Agents. The Orchestration Agent is the default interface for the human leader on that team. It receives goals, clarifies intent, decides what work is required, delegates to specialists, tracks dependencies, requests evaluation, and synthesizes the final result.

Specialist Agents have narrower jobs. A Revenue team might include a Pipeline Analyst, a Sales Enablement Agent, an Outbound Strategist, and a Deal Review Agent. An Operations team might include a Process Analyst, a Reporting Agent, a Delivery Coordinator, and an Administrative Agent. A Leadership team might include a Stakeholder Advisor, a Decision Quality Agent, a Delegation Advisor, and a Team Effectiveness Analyst.

The architecture does not prevent the human from talking directly to a specialist. Direct communication is useful when the founder wants to work deeply with one function. But the product is designed so the founder does not need to coordinate specialists manually. The orchestration agent exists precisely to remove that management burden.

> *All agents are accessible. One agent is accountable for coordination.*

IV. Multi-Team Topology: Bounded Contexts and Event-Driven Coordination

This section describes how multiple team-scoped applications coordinate to form a single AI organization. It is the architectural core of this document and the section a technical evaluator should read most carefully.

One Application Per Team

Each executive team is implemented as its own application, deployed in its own Lovable project. In the current model this yields four applications: Revenue, Operations, Leadership, and AI Executive. Each application owns its team's orchestration agent, its specialist agents, its team-specific UI for the human executive, and its team-specific persistent state.

The reasons to partition by team rather than build one monolithic application are conventional to anyone who has architected service-oriented systems, but worth stating explicitly:

- Blast radius containment. A failure or misconfiguration in one team does not degrade the others.
- Independent evolution. Each team can iterate on its agents, prompts, skills, and UI without coordinating a release across the whole organization.
- Clean transferability. A client can adopt teams incrementally and can, if necessary, retire or replace a team without unwinding the entire system.
- Legibility. The team boundary in the architecture matches the team boundary in the client's mental model of their business.

Two Coordination Substrates, Not One

Cross-team coordination happens through two substrates that play distinct roles. Conflating them is a common mistake in multi-agent architectures and is worth avoiding by design.

The event bus carries verbs. It transmits things that happened: a qualified opportunity, an approved decision, a completed onboarding, a scheduled review. Events are transient. They are the medium through which one team's work triggers another team's response.

The shared brain carries nouns. It stores what is true about the organization: the client profile, current strategy, key metrics, prior decisions logged by the human executive, cross-team commitments, and other organizational facts. The shared brain is persistent and queryable. It is the memory each team reads to understand context before acting.

> *Every architectural decision about cross-team behavior should be traceable to one of these two substrates. If a team needs to know something durable about the organization, it reads the shared brain. If a team needs to trigger action in another team, it emits an event.*

Inngest as the Event Bus

Inngest provides the event and durable workflow substrate. In the Lovable ecosystem, an Inngest connection is workspace-level rather than per-project, which means one Inngest workspace can serve all four team applications through a shared event key and signing key. This is precisely the coordination pattern the architecture requires. All four team apps publish to and subscribe from the same event stream without any application needing to hold a reference to another application's internals.

The event contract between teams is the interface. Each event has a defined name, a defined payload schema, and a documented set of producing and consuming teams. This event catalog is the coordination contract of the AI organization. It is deliberately kept small, versioned, and reviewed. New cross-team events are added intentionally rather than accreted casually.

The Cross-Team Handoff Pattern

When one team needs another to act, the pattern is always the same. The originating team's orchestration agent decides that a cross-team action is required. It writes any necessary durable state to the shared brain. It then publishes an event through Inngest. The sibling team's orchestration agent, subscribed to that event, receives it, consults the shared brain for context, decides how to respond within its own team, delegates to specialists, and produces a response either as further shared-brain state, as a return event, or as a notification to the human executive.

Direct agent-to-agent calls across team boundaries are prohibited by design. A Revenue specialist does not call an Operations specialist. If Revenue needs Operations to act, Revenue's orchestrator emits an event and Operations' orchestrator decides how to respond. This preserves the bounded context of each team and the human-legible chain of accountability that a human executive relies on to supervise the organization.

Conceptual Topology

The overall topology in one sentence: four team applications, each with its own orchestration agent and specialists, arranged around a shared brain and connected by an event bus, with a human executive as the accountable decision node at the center. A reference diagram accompanies live partner discussions.

V. The Shared Brain

The Shared Brain is the organization's persistent intelligence layer. It is not another chatbot. It is a structured store of company-specific context that agents can retrieve when performing work and that captures the outputs of that work over time.

At the content level, the shared brain may include company strategy, products and services, customer profiles, leadership priorities, organizational structure, terminology, approved policies, meeting decisions, key metrics, prior artifacts, lessons learned, and current initiatives. The shared brain combines structured data with curated documents and persistent summaries rather than becoming an uncontrolled dumping ground.

Operational State vs. Organizational State

Operational state is state a team needs to perform its own work. It lives inside that team's application. Organizational state is state that other teams need to understand what is true about the client and the engagement. It lives in the shared brain. The test is straightforward: does more than one team need to read it, or does the human executive need it to make cross-team decisions? If yes, it belongs in the shared brain.

Human Decisions Are First-Class Content

When the human executive approves, rejects, or modifies a proposed action, that decision is logged to the shared brain with the context that produced it. Sibling teams inherit these decisions without re-asking. This is the mechanism by which human judgment scales across the organization rather than becoming a repeated bottleneck at every team boundary.

Tenant Scoping Is a First-Class Schema Concern

Every shared brain record is scoped to a tenant. Tenant isolation is enforced at the database layer through row-level security or equivalent authorization, not at the application layer. This ensures that no agent, in any team application, can read or write to another client's organizational memory even in the presence of application-level errors.

VI. Skills: IP as Executable Capability

Agents need more than role prompts. They need procedural knowledge. Integral uses the term Skill for a reusable instruction set that describes how a particular kind of work should be performed. A skill defines the required inputs, the sequence of reasoning, the expected output, important constraints, and the criteria used to determine whether the work is complete.

Skills convert institutional knowledge into executable capability. A methodology from a book chapter becomes a skill. A checklist a founder uses in their head becomes a skill. A partner's playbook becomes a skill. Once encoded, the skill can be invoked by any authorized agent, versioned over time, evaluated against real outputs, and adapted to a specific client without rebuilding the agent that uses it.

Skills can be team-scoped or shared across teams. A team-scoped skill lives inside a single team's application. A shared skill is published centrally and referenced by multiple teams. The decision follows the same test used for state: if more than one team needs the skill, it is shared; otherwise it lives inside the team.

VII. Evals: Defining What Good Work Means

An autonomous system is not trustworthy merely because it can produce output. It must be able to determine whether that output meets an agreed standard. Evals provide that quality-control layer.

An eval may be deterministic, such as checking that a report contains all required sections, or judgment-based, such as scoring strategic coherence, evidence quality, clarity, relevance, or brand alignment.

Evals may also test business constraints: whether an action exceeds authority, whether a recommendation is supported by evidence, or whether a deliverable is appropriate for a particular stakeholder.

In human terms, evals are performance standards. A client needs to know what the AI employee is expected to do, how often it must do it, what quality level is acceptable, and which mistakes are unacceptable. The technical system translates those expectations into checks. If work fails, it is repaired, escalated, or stopped. If work passes, it is delivered or moved to the next step.

VIII. Loops: Recurring and Improving Work

A loop is the mechanism that allows an AI system to continue working instead of stopping after one response. Integral distinguishes three practical loop types.

The Artifact Quality Loop follows Generate, Evaluate, Repair, and Re-evaluate until the output meets the standard or reaches a stopping rule.

The Workflow Completion Loop verifies each step in a larger process before moving forward, ensuring that a multi-step delivery is coherent end to end.

The Performance Improvement Loop operates over real-world time. It establishes a target, observes current performance, chooses an action, measures the result, stores the learning, and chooses the next action. This is how an agent team improves as it operates.

For SMB customers, these technical loops are presented as recurring responsibilities, routines, reviews, and improvement cycles. Agent teams become more like employees than tools: they have jobs, routines, and standards.

IX. Durable Execution and the Role of Inngest

Not every loop can or should run inside one web request. Some work takes minutes. Some waits for human approval. Some must resume tomorrow. Some runs every Monday. Some depends on external systems that may fail temporarily. Lovable Cloud and Supabase Edge Functions handle normal application actions, AI calls, database writes, and short scheduled jobs. For work that must survive time and failure, Integral uses Inngest as a durable workflow substrate.

Inngest is not the intelligence. It is the execution fabric. The agent still reasons. The skill still defines the procedure. The eval still defines quality. The shared brain still stores organizational context. Inngest makes it practical for the workflow to wait, resume, retry, and continue reliably over time.

Inngest plays two roles in this architecture: durable workflow runtime for background jobs and scheduled tasks within a team, and event bus that carries cross-team coordination messages between team applications. Keeping these two roles conceptually distinct in design conversations helps avoid subtle bugs at team boundaries.

X. The Orchestrator-to-Human Communication Contract

The phrase "human in the loop" is easy to say and hard to implement well. This section describes the contract between each team's orchestration agent and the human executive it serves. This is the mechanism that makes human governance technically legible rather than philosophical.

When an Orchestrator Acts Autonomously

An orchestration agent acts autonomously when the requested work falls within a defined authority envelope: standard operating procedures, low-consequence actions, information gathering, and analysis. Within this envelope, the orchestrator delegates to specialists, runs loops, produces artifacts, and reports results to the human on a cadence rather than per-action.

When an Orchestrator Escalates

The orchestrator escalates to the human on four conditions: an action exceeds a defined authority threshold, the situation is ambiguous in a way that could produce a materially wrong outcome, the situation is novel and not covered by an existing skill or precedent, or a cross-team commitment is being made on behalf of the executive. Each escalation carries the context needed to decide.

How Human Decisions Propagate

When the human decides, the decision is logged to the shared brain along with the context that produced it. Sibling teams inherit the decision. If the executive approves a new pricing policy in a Revenue conversation, the Operations orchestrator sees that policy the next time it consults the shared brain. No one has to re-explain it.

Cadence Design

Different orchestrators surface information on different cadences. A Revenue orchestrator may surface a daily briefing. An Operations orchestrator may surface a weekly status. A Leadership orchestrator may surface event-driven interrupts before key stakeholder conversations. Cadence is a design decision made per team, tuned to the executive's working rhythm, and adjustable over time.

Why This Beats Fully Autonomous Agents for SMB Executive Work

Fully autonomous agents optimize for throughput. Executive work optimizes for judgment. The value an SMB executive brings to their business is disproportionately in the decisions they make, not the tasks they execute. An architecture that concentrates the executive's attention on decisions, while delegating execution to a coordinated agent organization, produces more value per hour of executive attention than an architecture that removes the executive from the loop.

> *The goal is not to automate the executive. The goal is to give the executive an organization worth leading.*

XI. Lovable as the Preferred Implementation Environment

Integral can implement this architecture on several modern application stacks. The architecture itself is platform-independent. Lovable is currently a strong default for SMB deployments because it compresses product design, frontend development, backend integration, and client administration into a single environment that a client can plausibly own and continue operating.

Lovable Cloud provides a built-in backend that includes database, authentication, storage, edge functions, and AI capabilities. This lets Integral deliver a real software application around the agent team rather than a hidden collection of scripts. The client sees users, tasks, artifacts, approvals, priorities, memory, routines, and performance through a legible interface.

The multi-team topology uses one Lovable workspace containing four projects, one per team, sharing a workspace-level Inngest connection and reading from a shared brain implemented on a Supabase-compatible data layer. Integral develops against a stabilized Golden Template for each team, clones it into a client engagement, customizes the client context, connects the required data and services, tests, and deploys.

XII. Security, Tenancy, and Data Boundaries

Application-Level Boundaries

Sensitive logic runs server-side rather than in the browser. Database permissions use row-level security or equivalent authorization so users and agents can only access records they are entitled to access. Secrets such as API keys are stored in protected server-side secret stores. High-risk tools are never exposed directly to the frontend.

Agent Permissions Mirror Human Authority

A Research Agent may read public sources and create internal artifacts. A Sales Agent may draft outbound messages but requires approval before sending. A Finance Agent may analyze data but is prohibited from executing transactions. The orchestration agent enforces these boundaries and routes exceptions to the human executive. Permissions are declared explicitly per agent per tool.

Client Tenancy

Every artifact, memory record, event, and workflow instance is scoped to a client tenant. Tenant isolation is enforced at the data layer, not the application layer, so that isolation is preserved even in the presence of application-level bugs. The shared brain schema treats tenant identity as a required key on every row.

Audit Trail

The system logs every agent action that touches an external system, every human approval or rejection, every cross-team event, and every material change to the shared brain. Audit records are tenant-scoped and retained on a schedule appropriate to the client's compliance posture.

XIII. Failure Modes and Observability

A system worth trusting is a system whose failure modes are named and handled.

Agent-Level Failures

Model errors, tool errors, and invalid outputs are the most common failures at the agent level. The Artifact Quality Loop is the first line of response. If the loop cannot produce a passing artifact within a defined stopping rule, the orchestrator escalates to the human with the failure context attached.

Workflow-Level Failures

Inngest handles transient failures through automatic retries with backoff. Persistent failures produce dead-letter records that surface to an operator queue. Long-running workflows can be paused, inspected, and resumed. Every failure has a defined path either to recovery or to human attention. No failure vanishes silently.

Cross-Team Event Failures

Missed subscribers, malformed payloads, and poison messages are handled explicitly. Every cross-team event has a defined schema and a defined set of consumers. Consumers that fail repeatedly on the same event surface as operator alerts. Events are replayable for the duration of the retention window.

Observability

Observability is layered. Agent decisions are logged with the reasoning that produced them. Orchestrator routing decisions are logged with the context that motivated them. Event flow across teams is visible in a single view. Human interventions are recorded alongside the agent activity they modified. An operator debugging the system should be able to answer three questions quickly: what happened, why did it happen, and what needs to happen next.

XIV. Deployment and Client Ownership

The client ownership model is explicit. Where practical, the production application lives in a Lovable workspace controlled by the client. The client controls billing, user access, production data, and infrastructure credentials. Integral receives the access needed to build, operate, and support the system during the engagement. At the end of the engagement, that access can be revoked without the client losing capability.

A completed engagement produces the following client-owned assets: the Lovable workspace containing the team applications, the Supabase-compatible database that holds the shared brain, the Inngest workspace that carries events and durable workflows, the agent configurations and skill libraries specific to the client, the eval definitions that describe acceptable work, and the documentation required to continue operating the system. Integral retains its own reusable architecture, Golden Templates, and general methodology as pre-existing intellectual property.

XV. Roadmap and Deliberate Limitations

Description what is live, what is next, and what is not yet solved.

Live Today

The 18-agent voice demo is deployed and accessible to prospects. The three-layer agent architecture (shared preamble, system prompt, knowledge file) is in production use across all 18 agents. The four-team model is designed and its team compositions are defined. The Golden Template pattern is proven across multiple builds.

Next

The multi-application production topology, with four team apps sharing an Inngest event bus and a shared brain, is the next major build. The cross-team event catalog is under active definition. The shared brain schema is being hardened for multi-tenant production use.

Deliberate Limitations

- **Cross-team event schema evolution is not yet fully versioned.** Breaking changes to a cross-team event currently require coordinated deployment across teams. A versioning discipline is planned but not yet in production.
- **Shared brain query patterns are not yet exhaustively indexed.** The schema is optimized for the queries the four teams issue today. Targeted indexing and query-layer caching follow as new patterns emerge.
- **Observability tooling is functional but not yet polished.** Logs and traces exist and are useful, but a purpose-built operator console for cross-team debugging is on the roadmap rather than in production.

These are engineering trajectories rather than architectural weaknesses. Naming them here is the point.

> *The architecture is complete. The implementation is in motion. The limitations named here are sequencing decisions, not design gaps.*

Appendix A. Glossary

Orchestration Agent	The team-lead agent in each team application. Interfaces with the human executive, delegates to specialists, and mediates cross-team communication.
Specialist Agent	A narrow-scope agent inside a team application. Performs a defined role such as analysis, drafting, research, or coordination.
Shared Brain	The persistent organizational memory layer. Stores what is true about the client and the engagement. Read and written by all teams under tenant scoping.
Skill	A reusable instruction set that defines how a particular kind of work is performed. Encodes methodology as executable capability.
Eval	A check that determines whether a piece of work meets an agreed standard. May be deterministic or judgment-based.
Loop	A mechanism that lets the system continue working over time. Types include Artifact Quality, Workflow Completion, and Performance Improvement.
Bounded Context	A team application's scope of responsibility. Teams coordinate only through shared substrates, not through each other's internals.
Event Bus	The substrate that carries cross-team verbs. Implemented on Inngest at the workspace level.
Durable Workflow	A workflow that survives time and failure. Implemented on Inngest for background jobs, scheduled tasks, and long-running processes.
Tenant Scoping	The discipline of isolating each client's data at the database layer through row-level security or equivalent authorization.
Three-Layer Architecture	The internal structure of each agent: a shared preamble, a role-specific system prompt, and a knowledge file.
Four Installations	The four executive teams in the Integral model: Revenue, Operations, Leadership, and AI Executive.
Golden Template	A stabilized reference implementation of a team application cloned and customized for each client engagement.
Universal Leadership Model	Integral's three-dimensional leadership framework underpinning the Leadership team's skills and agent behavior.
Cells	The nine sub-functions within each team's operating framework.

Appendix B. Reference Topology

The topology described in Section IV in prose form:

At the center sits the Human Executive, the accountable decision node. Arranged around the executive are four team applications: Revenue, Operations, Leadership, and AI Executive. Each team application contains one Orchestration Agent and several Specialist Agents. All four team applications read from and write to the Shared Brain, which holds tenant-scoped organizational memory. All four team applications publish to and subscribe from the Event Bus, implemented on Inngest at the workspace level. The Shared Brain and the Event Bus are the only paths of cross-team coordination.

This topology makes three properties visible at a glance: bounded contexts (each team is a distinct application), event-driven coordination (all cross-team communication flows through the bus), and organizational memory (a single shared brain accessible to every team). Together these three properties are the architecture.

INTEGRAL CORPORATION
Client-Owned AI Organizations for SMB Founders

integralexecutive.com