InterSystems IRIS

The Real-Time Data Platform for Agentic AI



Safely Connecting Agents to Enterprise Execution

A Technical Architecture Perspective for CTOs,
Chief Architects, and Engineering Leaders

InterSystems®
Creative data technology

Contents

Executive Summary 3

1. From Intelligent Responses to Operational Participation 5

2. The Anatomy of a Production Agentic System 6

3. Operational Context, Authoritative State, and Execution Control 9

4. The Missing Layer in Enterprise Agent Architecture 12

5. InterSystems IRIS as the Operational Execution Layer 14

6. Agentic, Deterministic, and Human Execution 16

7. Security, Governance, and Observability 17

8. Implementation Patterns 19

9. Reference Architecture 20

Conclusion: Connecting Intelligence to Execution 22

Executive Summary

The first generation of enterprise generative AI focused primarily on knowledge work: answering questions, summarizing information, generating content, and making enterprise knowledge more accessible. These systems informed work rather than participating in it. Organizations now expect AI systems to investigate incidents, coordinate activities, invoke enterprise services, escalate exceptions, request approvals, and contribute directly to operational outcomes.

This shift is often described as a move from answers to actions. That description is incomplete. Earlier automation could already take action through predictive models, rules engines, robotic process automation, APIs, and workflow systems. Agentic systems change the architecture because they can introduce model-driven reasoning into processes that have traditionally depended on deterministic execution. An agent may interpret an objective, assemble context, select tools, determine intermediate steps, revise a plan, and choose among available actions as conditions change.

That flexibility creates value, but it also increases the importance of the surrounding operational architecture. Production agentic systems need more than model intelligence, retrieved data, session memory, and tool access. They need current operational context, authoritative state, identity-aware access, bounded business capabilities, durable process state, deterministic controls, human approval where required, recovery from failure, and evidence of what was authorized and completed.

Models provide reasoning. Agent frameworks manage planning, model interaction, session state, and tool selection. Enterprise applications and operational platforms remain responsible for authoritative data, business rules, transactions, workflows, permissions, and committed outcomes. The task is to connect these responsibilities without transferring operational authority to the model or the agent framework.

This paper describes the layer that performs that role: a real-time data platform for agentic AI. It gives agents governed access to current operational context, enterprise capabilities, and durable processes. It complements models, agent frameworks, lakehouses, automation tools, and enterprise applications rather than replacing them.

InterSystems IRIS® fulfills this role. It provides operational data management, multi-model access, vector search, transactional-analytic processing, interoperability, durable workflows, security, and auditability all as part of a unified platform architecture. It also makes these capabilities available to native and external agents through governed interfaces, while keeping authoritative controls with the data, tools, workflows, and connected applications responsible for execution.

What distinguishes InterSystems IRIS is where these capabilities reside. Rather than distributing data, integration, and process state across separate products, InterSystems IRIS brings authoritative operational data, vector search, interoperability, and long-running workflow state together in one transactional platform. For IRIS-managed resources, this means fewer handoffs where state can diverge or controls can be bypassed: the conditions that determine whether an action is valid are checked where that state is maintained, under one governance model.

The central thesis is straightforward: models can reason, and agent frameworks can orchestrate that reasoning, but production agentic systems still require an operational foundation that connects probabilistic reasoning to authoritative state and reliable execution. As agents move into mission-critical processes, that foundation becomes a core component of enterprise AI architecture.

1. From Intelligent Responses to Operational Participation

For decades, enterprise architecture maintained a relatively clear division of responsibility. Operational systems processed transactions, maintained authoritative records, enforced business rules, and supported business processes. Analytical systems organized information for reporting, forecasting, optimization, and decision support. Software executed predefined logic. People interpreted information, exercised judgment, and initiated action.

The first generation of generative AI largely preserved this division. Agentic AI changes it: AI systems now take part in the processes themselves, not just in the information surrounding them.

Established Disciplines Applied to a New Execution Model

The operational requirements surrounding agentic systems are not entirely new. Enterprises have long needed transaction management, workflow state, recovery, identity, approvals, interoperability, observability, and governance across distributed systems, integration platforms, workflow engines, rules systems, and business applications have addressed these requirements for decades. What is new is a participant whose next step is not fixed in advance. A workflow follows transitions set by a process designer, whereas an agent chooses its path at run time based on how it interprets the objective.

The architectural challenge is therefore not to replace established operational disciplines. It is to integrate model-driven interpretation and planning with the mechanisms already responsible for reliable enterprise execution. The surrounding architecture must ensure that this flexibility operates safely within explicit boundaries.

Operational Participation Is Different from Tool Use

An agent calling an API does not necessarily constitute meaningful participation in a business process. A tool call may retrieve a record, create a ticket, send a message, or update a field. A business process may extend across multiple systems and participants, continue for hours or days, require several approvals, and encounter conditions that were not known when it began.

Consider an agent assisting with a disputed transaction. It may need to verify the customer, establish the operating identity, retrieve current account and transaction information, review previous interactions, determine the dispute status, apply contractual rules, identify permitted remediation options, request approval, update the system of record, notify the customer, and preserve evidence of the outcome.

Each step may be exposed as a tool. The business outcome, however, depends on the coordinated execution of the process as a whole. Mission-critical architectures typically require mechanisms to establish which steps completed, which decisions remain pending, whether the underlying state changed, and how to proceed after a timeout, duplicate request, denial, or partial failure.

Risks Emerge at Process Boundaries

In an isolated demonstration, an agent may retrieve information, invoke a tool, and return a successful result. Production environments introduce normal distributed systems conditions: data changes between retrieval and execution, callers lack permissions, records are modified concurrently, downstream services time out, one step succeeds while another fails, repeated requests create duplicate effects, or human approval is delayed.

An operational agent architecture must address questions such as:

- What is the authoritative state of the process?
- Which identity is requesting the action?
- What data and tools can that identity access?
- Which rules and policies apply?
- What approvals are required?
- Can the request be retried safely?
- How will partial completion be detected and managed?

A more capable model may improve planning, interpretation, and tool selection. It does not by itself provide the required capabilities for transaction management, workflow recovery, authorization, audit evidence, or cross-system process state.

The Standard for Enterprise Action

For agentic participation in mission-critical operations, any action must be authorized, valid, consistent, recoverable, observable, and auditable.

Authorized: The request executes under an authenticated identity with appropriate access to the data, tool, and operation.

Valid: The action complies with current rules, policies, process state, contractual obligations, and approvals.

Consistent: The action preserves the integrity of data and process state when several participants or systems may act concurrently.

Recoverable: Timeouts, retries, duplicate requests, and partial completion do not leave the process in an unknown or invalid condition.

Observable: The end-to-end action can be followed by correlating its tool invocation, authorization, workflow transitions, downstream calls and resulting changes.

Auditable: Establish who or what initiated the request, which identity and controls were applied, what completed, and what outcome was recorded.

The defining transition is not simply from answers to actions. It is from AI operating within narrowly defined tasks to AI participating dynamically and safely within production operational processes.

2. The Anatomy of a Production Agentic System

Much of the discussion surrounding agentic AI usually begins with model capability. Models are evaluated according to their ability to reason, plan, generate code, interpret multimodal information, follow instructions, and use tools. These capabilities materially affect performance, but they are not sufficient.

The success of a production agentic system also depends on the quality and authority of its context, the precision of its tool contracts, the identities under which requests execute, the controls governing access and action, the mechanisms preserving process state, and the ability to detect and recover from failure.

What appears to a user as a single agent is better understood as a set of cooperating architectural responsibilities. These responsibilities do not necessarily correspond to separate products. A single platform may perform several, while a distributed architecture may assign them across multiple systems.

Reasoning and Planning

The reasoning layer interprets objectives and determines how an agent should respond. It may classify requests, decompose objectives, compare alternatives, generate structured outputs, identify missing information, select tools, evaluate intermediate results, and revise a plan.

This layer is intentionally probabilistic. Its value comes from working with ambiguity and tasks that cannot be fully encoded in advance. A model-generated decision may inform an operational action, but it should not be treated as proof that the action is authorized, valid, or complete.

Agent Orchestration

The agent layer coordinates model requests, session state, retrieval, model selection, tool discovery and invocation, multi-agent interaction, and model-oriented safeguards. Some agent platforms also add features such as task checkpointing, tool gateways, approval prompts, and tracing.

These features are oriented to the agent's own activity: its reasoning, sessions, and tool calls. They do not make the agent layer authoritative for the business. Whether a downstream operation committed, whether a required approval is valid under current policy, or whether a long-running business process reached a valid final state is determined by the platforms and applications that own the business data, rules, and transactions.

Context Acquisition and Assembly

Context may be assembled from operational databases, documents, event streams, APIs, vector search, knowledge graphs, user input, conversation history, and workflow systems. Retrieval, filtering, ranking, transformation, and summarization make the information usable by the model.

The context supplied to a model is a representation of the relevant environment, not the environment itself. Conditions that determine access or validity must therefore be evaluated again by the responsible system at the point of action.

Tool and Capability Mediation

Agents require controlled mechanisms for interacting with enterprise capabilities. A tool may expose a query, business method, application service, API operation, workflow, calculation, transactional action, or request for human approval.

A production tool should represent a bounded business capability rather than unrestricted system access. Its contract should define permitted inputs, returned outputs, operating identity, access requirements, validation rules, and more. The agent can choose among approved capabilities but it should not determine the security model governing them.

Operational Process Execution

Operational process execution manages persistent business state, deterministic execution, retries, exceptions, transaction management, human approvals, recovery, and process history.

Its core state consists of business records, workflows, transactions, operational outcomes, and the authoritative record of completed and pending work. These are materially different from agent session memory. A durable process should be able to resume from an authoritative record even when the agent, model, runtime, or downstream service fails.

Security, Policy, and Evidence

A production architecture must establish who or what is making each request, what that identity may do, and what evidence shows the outcome. Operational platforms enforce controls over the resources they manage, and connected applications remain responsible for their own capabilities. Model traces show what an agent attempted; workflow, transaction, and application records show what actually occurred. A tool call trace is not proof that the business action completed. Chapter 7 covers these controls in detail.

Designing the Relationship Between Agents and Processes

For many mission-critical workloads, a safer pattern is to allow agents to participate within governed processes rather than making them solely responsible for process control. Agents contribute interpretation and adaptability, deterministic components provide repeatability and precision, workflow services coordinate state and recovery, and people provide judgment where risk or policy requires it. Chapter 6 describes how these forms of execution divide the work.

From Demonstration to Production

A demonstration can succeed when an agent selects a tool and receives a plausible result. A production system must also handle the conditions described above, along with wrong tool selection and runtime restarts. In each case, the surrounding architecture, not the model, determines what happens next.

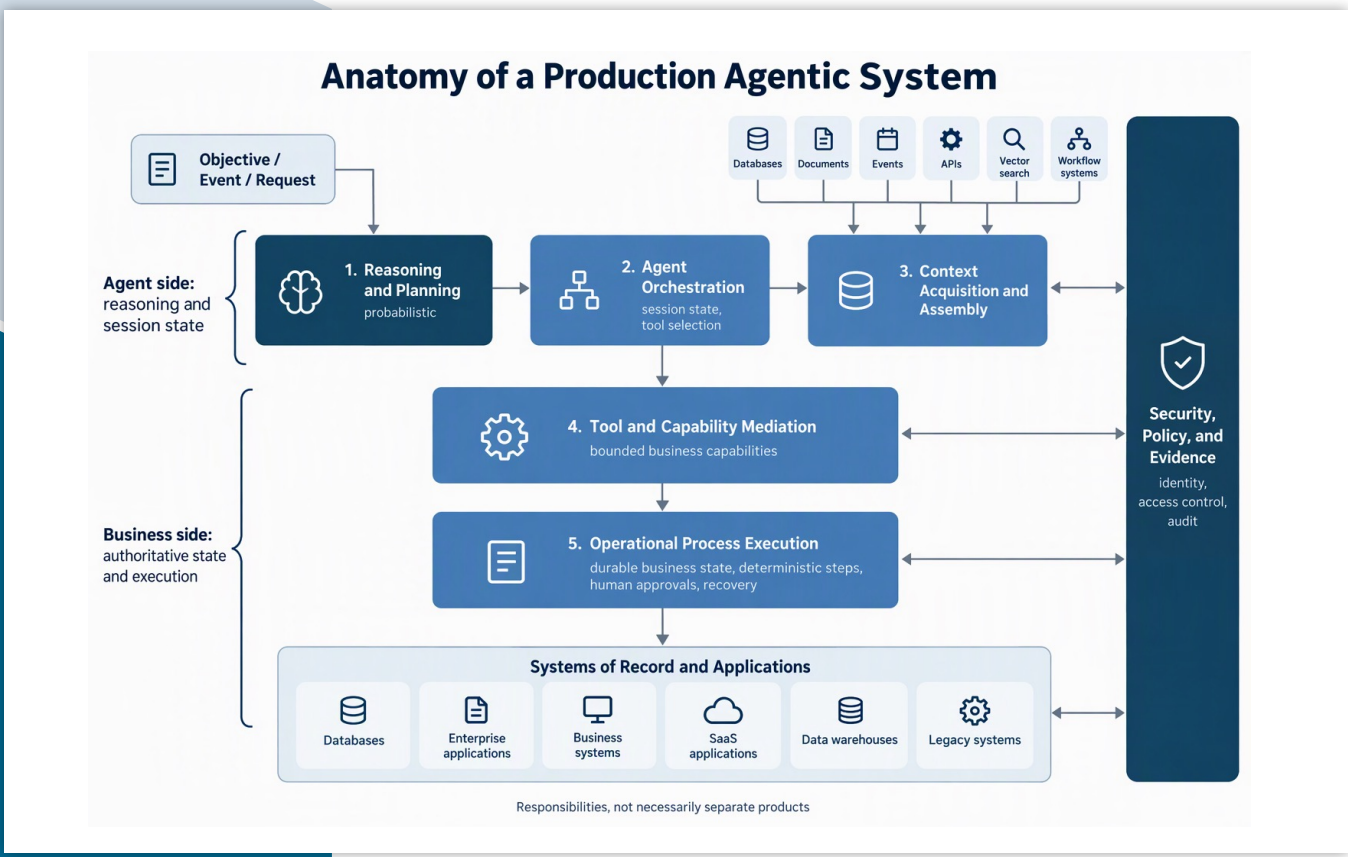


Figure 1. The cooperating responsibilities of a production agentic system. They do not necessarily correspond to separate products.

3. Operational Context, Authoritative State, and Execution Control

One of the most common assumptions in enterprise AI is that an agent becomes more effective as it gains access to more data. This assumption emerged from the first generation of generative AI applications, where retrieval-augmented generation addressed gaps in the model's knowledge of proprietary or recent information.

Embeddings, vector search, semantic retrieval, knowledge graphs, and context management remain essential. But an agent participating in business operations faces a different problem than a system answering a question. It needs the current, authoritative, and task-relevant conditions required to evaluate and perform an action safely. That requirement is usually described as operational context.

Data Does Not Equal Operational Context

Data describes aspects of the business. Operational context establishes how those aspects relate to the current decision or proposed action.

In a healthcare prior authorization process, patient records and clinical documents do not by themselves establish the request status, completed reviews, applicable payer requirements, missing information, task ownership, outstanding approval, permitted next action, or whether circumstances changed since the case was evaluated.

The same pattern appears elsewhere. Transaction data does not provide complete context for a fraud investigation. Inventory records do not establish the complete context for a supply disruption. Customer history does not establish the complete context for resolving an escalation issue. In each case, the agent requires information, process state, identity, policy, and execution conditions.

A Working Definition of Operational Context

Operational context is the current, task-relevant combination of authoritative data, events, process state, identity, policy, and execution conditions required to evaluate and perform an action safely.

Current: The context reflects conditions relevant at the time of decision and action. Freshness reduces the risk of outdated data, but does not by itself establish correctness, completeness, or authority.

Task relevant: The agent receives information required for the task rather than access to every related record or system.

Authoritative: Critical state comes from or is validated against the platform or application responsible for maintaining it.

Combined: Safe action depends on multiple forms of context evaluated together.

Action oriented: Operational context helps determine what is allowed to occur next, subject to controls and validation applied by the responsible systems.

From AI-Ready Data to Agent-Ready Data

The emergence of generative AI created renewed focus on AI-ready data. The term generally refers to data that can be reliably and responsibly used by AI systems. It should be accessible, discoverable, sufficiently current, appropriately structured, and governed for its intended use. Its quality, meaning, provenance, and access controls should be understood well enough to support retrieval, grounding, analytics, model development, and other AI workloads.

Depending on the use case, AI-ready data may encompass operational and transactional data, documents, analytical and historical data, master and reference data, metadata and lineage, semantic definitions, embeddings and vectors, identity and access policies, and governed retrieval interfaces.

These capabilities remain necessary. Poor quality, incomplete, stale, or insufficiently governed data can undermine an application regardless of model sophistication.

As AI moves from generating and retrieving information to participating in business processes, the requirements expand. AI-ready data remains necessary, but production agentic systems require agent-ready data.

Agent-ready data supports more than model consumption. It remains connected to the identities, policies, business logic, workflow state, permissions, and execution mechanisms that determine how the data may be used. Authoritative records can remain in operational platforms, applications, workflow systems, identity services, and other systems of record, provided the task relevant information and capabilities are available through governed interfaces when the agent reasons or acts.

Agent-ready data is still only part of the production architecture. An agent participating in a business process also requires current workflow state, rules, entitlements, permitted operations, process history, approvals, transaction boundaries, recovery, and exception handling.

For production systems, the progression is AI-ready data to agent-ready data to agent-ready operational context, which combines usable, governed data with current process state, business rules, identities, permissions, workflow conditions, and execution controls.

The Dimensions of Operational Context

Data context: Records, transactions, documents, vectors, and application state. Different access models may operate over the same governed information.

Event context: Recent changes across applications, services, partners, and users. Events indicate that something occurred; the responsible system often maintains the resulting authoritative state.

Workflow context: The current stage, completed steps, pending approvals, deadlines, and available transitions. For long-running work, this state must persist independently of a model session.

Security context: Authenticated identities, roles, data entitlements, and operation-specific permissions. Security context must be enforced, not merely stated in a prompt.

Business and policy context: Eligibility rules, contracts, regulations, risk thresholds, and approval limits. Model interpretation of a policy does not replace deterministic enforcement or human review where required.

Execution context: Service availability, transaction boundaries, dependencies, and recovery state. An action can be logically appropriate but operationally unsafe.

Context Does Not Confer Authority

Retrieved context includes search results, documents, vector matches, cached records, summaries, knowledge-graph results, previous messages, and tool responses. It helps the model interpret the situation and propose a course of action. Authoritative operational state includes conditions maintained by the responsible system: current transaction status, active workflow stage, latest account balance, inventory position, recorded approval, current entitlement, and committed outcome.

Authoritative state may reside in the operational layer, a connected application, a workflow service, an identity platform, or another system of record. The requirement is not centralization. It is governed availability when decisions and actions occur. Once information has been copied, cached, summarized, or inserted into model context, it should not be assumed to remain current.

An agent does not become authoritative because it has access to context. The authoritative source for an account balance, a workflow decision, or an entitlement remains the system responsible for it. An operational platform may mediate access, apply its own policies, maintain process state, and coordinate calls across multiple environments, but it does not automatically become authoritative for every connected resource. A sound architecture makes these boundaries explicit and preserves them throughout execution.

Context Must Be Evaluated at Several Points

Reasoning time: The agent receives task-relevant information to interpret the objective and propose a course of action.

Access time: The platform determines whether the operating identity may retrieve the information or invoke the capability.

Authorization time: The responsible system evaluates the action against current identity, record state, process stage, rules, and approvals.

Execution time: The responsible system verifies that conditions remain valid and attempts to commit the action within the applicable transaction or workflow boundary.

After execution: The architecture records the result, updates process state, and correlates the outcome with the initiating request, identity, approval, and downstream changes.

4. The Missing Layer in Enterprise Agent Architecture

Most enterprise agent architectures include several familiar elements. Foundation models provide language understanding and reasoning. Agent frameworks manage model interaction, memory, planning, and tool selection. Data and analytical platforms provide information for training, retrieval, and analysis. Enterprise applications and systems of record maintain the data and capabilities through which the organization operates.

What is often underdeveloped is the operational layer connecting these elements. Enterprises already possess databases, integration technologies, workflow engines, API platforms, observability tools, and business applications. What they often lack is a way for these capabilities to work together as a coherent, safe path from reasoning to authorized execution.

Where the Gaps Appear

Agent frameworks are built around the agent itself. The harder problems appear at the boundaries between the agent layer, the data platforms that supply context, the workflow systems that hold process state, and the applications that commit outcomes. When these responsibilities are distributed across separate products, each handoff creates gaps that are easy to overlook during a proof of concept:

- **Stale or incomplete context:** Agents often reason over documents, cached records, analytical copies, or retrieved summaries held in a different system from the authoritative record. These may be adequate for explanation but insufficient for action when current process state, transaction status, or entitlement determines validity. Each copy between the operational source and the agent adds delay and another opportunity for divergence. Adding more retrieval does not solve the problem; the architecture needs a route to authoritative state and a way to revalidate conditions at execution.
- **Fragmented governance:** Agent-level controls may govern which tools an agent can call, but data entitlements, business rules, and application permissions are typically managed elsewhere. When each product enforces its own portion, it becomes difficult to establish which identity acted, which policies applied, and which system made the final authorization decision. Agent-level controls do not replace authorization at the data, workflow, and application layers.
- **Workflow blindness:** An agent can track its own task without knowing the state of the business process around it: that a review already occurred, an approval is outstanding, a deadline is approaching, or another participant owns the next step. Without a connection to authoritative process state, the agent can repeat work, bypass required transitions, or treat a long-running case as a new task.
- **Agent state used as business state:** When agent checkpoints or memory stand in for the business record, process state becomes tied to a particular model, framework, or runtime. Business state must remain available independently of the model and agent framework, surviving model changes, restarts, delayed events, and downstream outages.
- **Tool sprawl and inconsistent contracts:** Tool gateways and protocols such as MCP make it easy to expose hundreds of APIs, methods, queries, and services directly to agents. Without business-level contracts, it can create risks around inconsistent naming, authentication, validation, error handling, and audit behavior. Agents benefit from bounded, task-oriented capabilities with explicit contracts.
- **Unreliable cross-system execution:** A process spanning several systems rarely shares one atomic transaction, and retrying a tool call does not by itself establish what each system committed. The process needs explicit handling for durable state, partial completion, compensation, escalation, and human intervention, none of which should be delegated to probabilistic model behavior.

- **Fragmented observability:** When agent traces and business records reside in separate products with separate identifiers, reconstructing what happened can become a challenge.

These gaps are not evidence that agentic AI is impractical. They indicate that agent deployments require the same rigor enterprises apply to other mission-critical distributed systems, extended to account for dynamic model-driven reasoning.

Agent frameworks are built around the agent itself. The harder problems appear at the boundaries between the agent layer, the data platforms that supply context, the workflow systems that hold process state, and the applications that commit outcomes.

The Gap Between Reasoning and Execution

An agent framework can decide that a capability should be invoked. It does not own the current business state, the business rules, the transaction boundary, or the authoritative process history. An enterprise application may own the authoritative record but have no awareness of the model's reasoning. An analytical platform may provide extensive information without managing the business process in which an action occurs.

The missing layer is therefore not another reasoning engine. It is an operational foundation that connects reasoning to authorized enterprise capabilities while preserving the semantics, state, controls, and evidence required by the business.

Responsibilities of the Operational Layer

- Assemble task-relevant operational context from current authoritative sources.
- Expose bounded data, tools, services, methods, and workflows rather than open-ended system access.
- Carry authenticated identities and apply access controls at the point of use.
- Coordinate agentic reasoning with deterministic logic, workflow state, and human approval.
- Maintain durable process state independently of a model session.
- Manage retries, timeouts, idempotency, exceptions, compensation, and recovery.
- Preserve authority boundaries across connected applications and systems of record.
- Correlate model activity with tool calls, authorization decisions, workflow transitions, and committed outcomes.

Defining a Real-Time Data Platform for Agentic AI

A real-time data platform for agentic AI provides the operational context, interoperability, governance, workflow participation, and execution services required to connect intelligent systems with enterprise operations.

The phrase real-time refers to more than speed. It describes the ability to work with operational conditions while they remain relevant to a decision or action. The required latency varies by process, but the architecture must avoid treating a delayed analytical copy as the source of truth when current state determines validity.

The phrase data platform remains important because data, state, transactions, security, and business logic are inseparable from operational execution. Agentic AI expands the data platform's role: data must be available not only for analysis and generation but also as part of a governed path through which agents participate in work.

A sound architecture assigns responsibilities deliberately. Models interpret and propose. Agent frameworks coordinate plans and tool calls. Operational platforms mediate access, maintain process state, execute governed logic, and coordinate workflows. Systems of record remain authoritative for the data and actions they own. Humans provide judgment and accountability where policy or risk requires them.

This separation allows organizations to change models or orchestration frameworks without recreating the underlying business controls in every agent implementation. It also enables the same governed capability to be used by conventional applications, human-operated systems, native agents, and external agents.

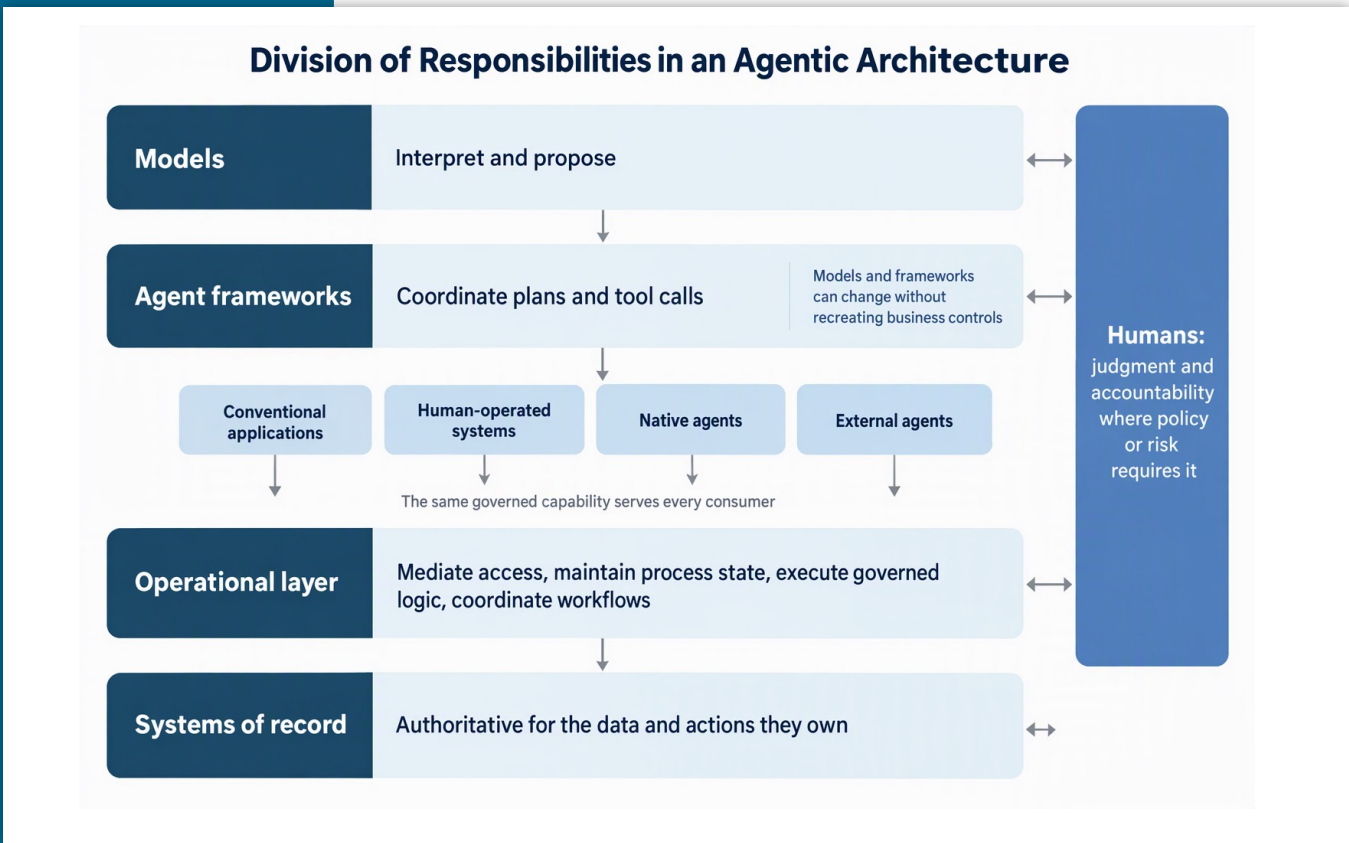


Figure 2. Division of responsibilities in an agentic architecture. Each participant keeps its own responsibility, and the same governed capabilities serve every consumer.

5. InterSystems IRIS as the Operational Execution Layer

InterSystems IRIS was not originally created in response to agentic AI. It was created to support operational applications and processes that require data management, interoperability, analytics, transaction integrity, workflow coordination, security, and high availability spanning distributed applications and data.

Those established capabilities align closely with the requirements of production agentic systems. IRIS can act as the operational layer through which agents safely access current data, use approved business capabilities, participate in workflows, and interact with connected systems under explicit controls.

Many of these capabilities are available individually elsewhere: vector search, databases, integration, workflow, governance, and analytics. The difference lies in where they run. Because they share one engine under common governance, an action on resources IRIS manages can be validated against current conditions at the point where those conditions are maintained, without passing through a chain of separately governed products. IRIS is also not tied to a single cloud or application suite and can be deployed on premises, in the cloud, or in hybrid environments, so it complements the broader AI, data, and application ecosystem rather than requiring organizations to standardize on one vendor's stack.

Agent frameworks manage the agent's reasoning and tool use. IRIS manages the business state those agents act on, with operational data, vectors, interoperability, and durable process state in the same transactional engine.

Consider an agent that proposes a credit adjustment on a customer account. In a distributed design, the agent reads a balance from a retrieval index or analytical copy, a workflow service sequences the approval, and the application commits the change through an API. Each hop adds delay, and the balance the agent saw may no longer be the balance at commit time. The design then needs revalidation, retry, and compensation logic spread across three products.

When the account data, the approval workflow, and the adjustment logic are managed through IRIS, validation and commit can occur in the same transaction against the same current state, and a failed condition rolls back cleanly. The agent still only proposes; the governed capability decides. This advantage applies only to resources IRIS manages. Where the account lives in a connected application, that application remains the authority, and IRIS coordinates the call and records the outcome.

Multi-Model Operational Data

IRIS is multi-model, not polyglot: data is held in a single representation and projected as relational, object, document, key-value, and vector views, without storing or mapping copies across separate engines. An agent can use the access model appropriate to the task while the underlying information remains connected to the operational environment in which it is maintained.

Vector similarity can identify semantically related content. SQL can apply exact filters and joins. Object and document access can preserve application structure. Key-value access can retrieve state efficiently. The benefit lies in combining these access patterns without fragmenting the data, security, and transaction model across specialized stores.

Native Vector Search Near Operational Data

IRIS stores vector representations alongside the records and content they describe. This supports semantic retrieval without requiring workloads to move operational data into a separate vector database. Retrieval can combine similarity with current attributes, permissions, business rules, and transaction state.

Keeping vectors close to source records can reduce synchronization pipelines and additional copies.

Real-Time Transactional-Analytic Processing

Agent decisions often depend on conditions that changed seconds earlier: a payment posted, inventory moved, a result arrived, a case advanced, or an approval completed. IRIS supports transactional processing and analytical access against current operational data within a common environment, allowing applications and agents to evaluate recent conditions without always waiting for a separate analytical copy.

Agents do not need direct access to transactions to benefit. Governed queries, methods, and services can shape the information presented to the agent while IRIS maintains the consistency and controls associated with the underlying operational state.

Integration Across Heterogeneous Systems

Enterprise processes rarely reside in one platform. IRIS integration and interoperability capabilities connect applications, databases, services, messages, events, devices, partners, and people through adapters, APIs, routing, transformation, business rules, and process orchestration.

Integration flows can receive events, validate and transform messages, enrich them with operational information, invoke services, route exceptions, and maintain process visibility. For agentic systems, this provides a managed path into existing operations rather than requiring the agent framework to coordinate each dependency directly.

Durable Process Execution

IRIS can maintain persistent workflow state and coordinate long-running processes involving deterministic logic, application calls, delays, retries, recovery, exceptions, and human approvals. Agents can be invoked for steps that benefit from interpretation or planning, while the operational process remains authoritative for sequencing and completion.

Process state is held in the same transactional engine as the operational data it concerns, so it is covered by the same transaction and recovery mechanisms. Integration flows queue messages persistently and track the state of each one, which allows in-flight work to resume after a restart or failover. The workflow resumes from its last committed state without asking the model to reconstruct what happened from conversation history.

Availability and Recovery

Because this layer sits in the execution path of agent-driven work, what happens during a failure is critical. In a distributed design, the database, the workflow service, and the agent runtime typically recover separately, and afterward they can disagree about what completed. With IRIS, process state is held in the same transactional engine as the operational data, so after a mirrored failover the backup already holds both the committed data and the committed process state. In-flight work resumes from that point.

Governed Business Capabilities

IRIS methods, queries, services, and workflows can be exposed as bounded capabilities with the explicit contracts described in Chapter 2. This allows agents to request approved outcomes without receiving direct access to internal databases or application code.

Calls to connected applications go through their supported interfaces, so those applications keep their own authorization and committed state, consistent with the boundaries described in Chapter 3.

AI Hub and External Agent Ecosystems

AI Hub, a set of native capabilities within IRIS, lets organizations build agents with the native agent framework in IRIS, bring an external agent framework of their choice, or use both together. AI Hub does not replace a customer's preferred models, agent frameworks, lakehouses, AI services, or enterprise applications.

Both kinds of agent use the same governed tool infrastructure. External agents reach it through Model Context Protocol (MCP), discovering and invoking the capabilities IRIS exposes, while native agents call it directly. Through MCP, IRIS-hosted agents can also use tools served by other systems. In every case, requests pass through the same identity, authorization, and audit controls.

An external framework continues to manage its own prompts, plans, model selection, sessions, and reasoning behavior. Whichever framework is used, IRIS provides the governed operational path through which approved enterprise capabilities are used, keeping business controls in one place as agent frameworks change.

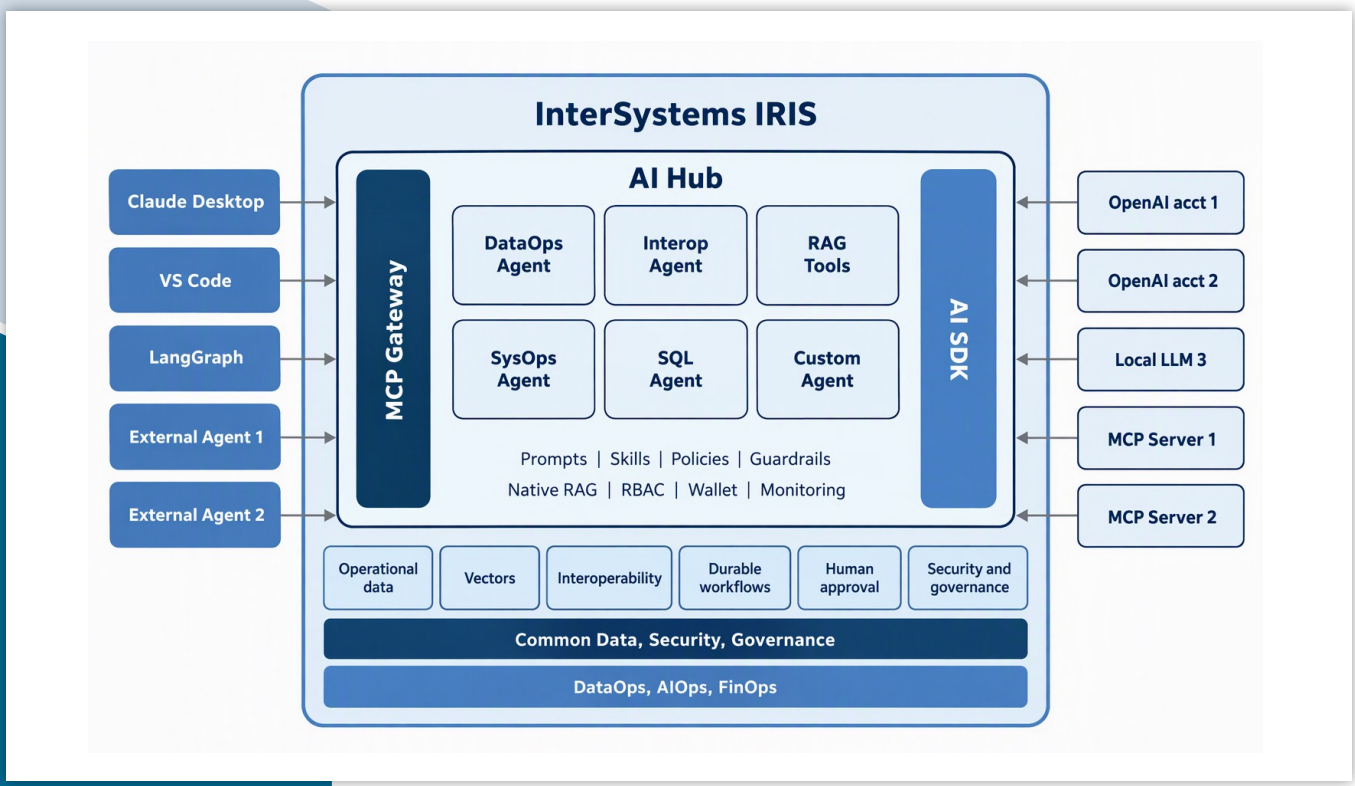


Figure 3. InterSystems IRIS AI Hub. External agents reach governed tools through MCP, native agents run inside IRIS, and both use the same security and governance.

5. Agentic, Deterministic, and Human Execution

Production enterprise operations combine several forms of decision making and execution. Agentic reasoning provides flexibility where interpretation and adaptation matter. Deterministic software provides precision and repeatability. Humans provide judgment, accountability, and exception handling. Most processes need all three, coordinated within one workflow.

Agentic Reasoning

Agents are valuable when the task involves unstructured information, ambiguous objectives, changing conditions, comparison of alternatives, or explanation. An agent can summarize a complex case, identify missing evidence, propose a response, classify an exception, or generate a plan.

Its output should be treated according to the consequence of the task. Low-risk outputs may proceed automatically after validation. Higher-risk recommendations may require deterministic checks, policy evaluation, or human approval.

Deterministic Execution

Deterministic components remain appropriate for fixed calculations, regulated rules, validation, transaction updates, routing, and system integration. They provide repeatability and make expected outcomes easier to test and audit.

Human Judgment and Accountability

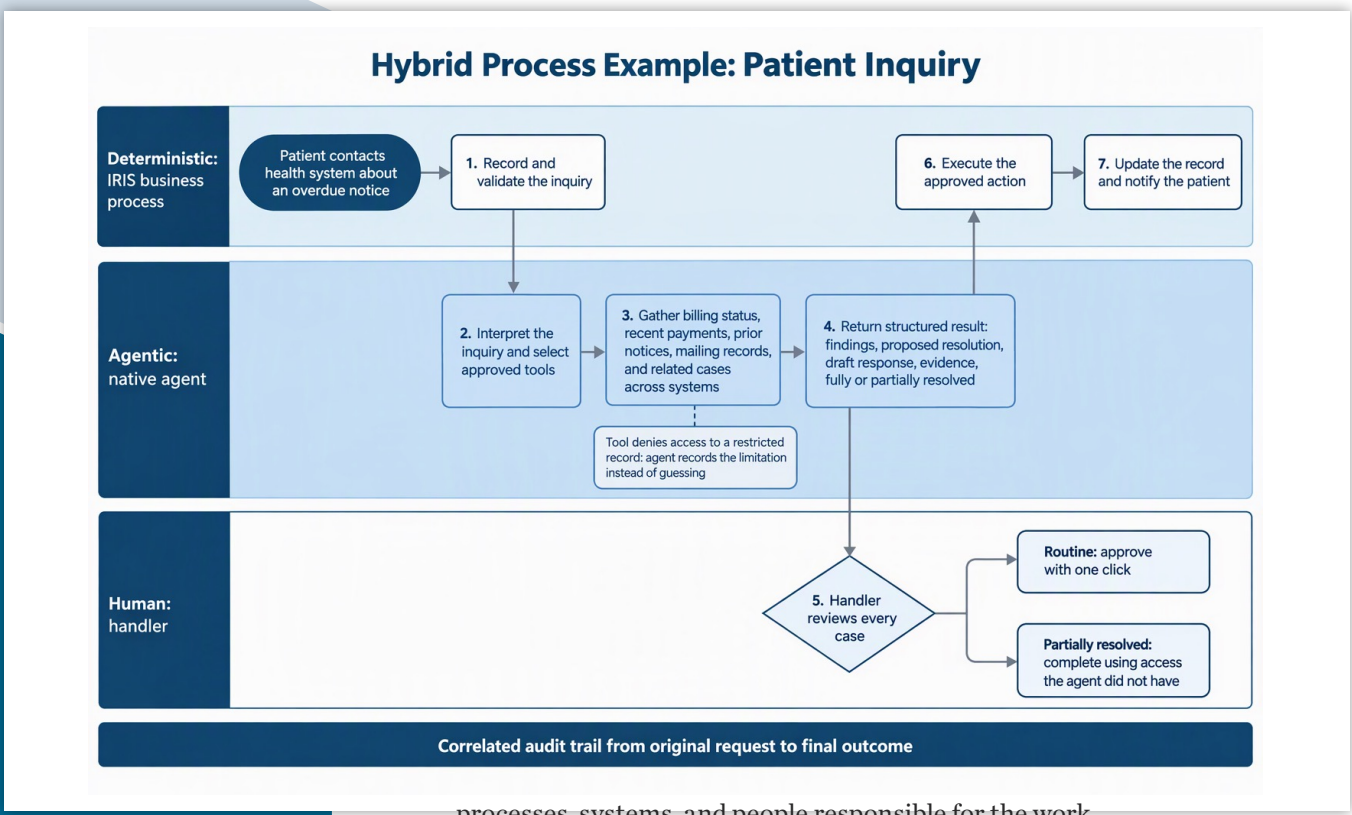
Human participation remains necessary where policy requires named accountability, consequences are material, facts are ambiguous, or human judgment cannot be reduced to a rule.

Human involvement should be designed into the process rather than treated as an informal fallback. The workflow should provide the reviewer with relevant context, the agent's recommendation, required evidence, available options, and a clear record of the final decision.

A Hybrid Process Example

Consider a healthcare patient inquiry. A patient contacts a health system about a 60-day overdue notice for a bill they never received. An IRIS business process records and validates the inquiry and passes it to a native agent. The agent interprets the inquiry and selects approved tools to gather current billing status, recent payments, prior notices, mailing records, and related cases across systems. When a tool denies access to a restricted record, the agent records the limitation rather than guessing. It returns a structured result with findings, a proposed resolution, a draft response, supporting evidence, and whether the case is fully or partially resolved. A handler reviews every case, approving a routine resolution with one click or completing a partially resolved case using access the agent did not have. The business process then executes the approved action, updates the record, notifies the patient, and preserves a correlated audit trail from the original request to the final outcome.

No single participant controls the entire process. Each contributes what it does best, and the durable workflow holds them together. Agents become participants in enterprise processes rather than replacements for the



processes, systems, and people responsible for the work.

Figure 4. A hybrid process example. The IRIS business process, a native agent, and a human handler each contribute what they do best within one durable workflow.

7. Security, Governance, and Observability

An agent driven process may retrieve data from one application, request approval in another, and update a third. Each system protects its own resources, but the process also requires controls across the sequence: who may initiate it, which identity is used, which approvals are required, how failures are handled, and how the final outcome is verified.

Identity and Delegation

Every agent request should run under an identity that can be authenticated, authorized, and audited, and the agent should hold no more access than its task requires. In IRIS, human and agentic access are governed in one model instead of in separate systems. When a process calls a connected application, IRIS can map that identity for the integration, and the application evaluates the credentials it receives.

For the resources it manages, IRIS provides controls that enterprises already rely on. Users and applications can authenticate through mechanisms such as LDAP, Kerberos, and OpenID Connect or OAuth 2.0. Authorization is role-based, with row- and column-level security available for data. Databases,

including their indexes, can be encrypted at rest, and connections can be protected with TLS in transit. System and application events are recorded in an append-only audit log that can be queried with SQL. Agents work within these same controls. These controls apply to resources IRIS manages; connected applications continue to enforce their own.

Authorization at the Point of Use

Authorization must be applied to the specific data, tool, or operation being requested. Permission to connect to an agent endpoint is not equivalent to permission to execute every capability behind it. Controls may depend on the caller, record, field, process stage, proposed action, approval state, and current business conditions.

Consequential actions should be revalidated at execution time because the underlying state may have changed since reasoning or retrieval occurred.

Bounded Capabilities and Least Privilege

Tools should expose the minimum capability required for the task. A narrowly scoped service that retrieves an approved customer view or submits a defined adjustment is easier to govern than unrestricted SQL, broad application credentials, or arbitrary code execution. Narrow tools are also easier for a model to use correctly: a specific, well-documented tool with clear inputs and outputs leaves the model less room for error than a generic one.

Approvals and Separation of Duties

When policy requires review, the process should create an explicit approval task and block the dependent action until a valid decision is recorded. The reviewer, evidence, decision, time, and resulting transition should become part of the process history. Where separation of duties applies, the agent or identity proposing an action should not also satisfy the independent approval requirement.

Transactions, Partial Completion, and Recovery

Transaction controls protect changes within their scope. Cross-system processes require explicit handling of partial completion because one transaction rarely spans every application. The workflow should record completed steps, apply retries deliberately, and define compensation, escalation, or human recovery when the process cannot complete automatically.

Observability Across Reasoning and Execution

Observability should correlate the initiating identity, model and agent activity, retrieved sources, tool calls, authorization decisions, approvals, workflow transitions, downstream requests, transaction results, exceptions, and final outcome.

Within IRIS, the interoperability layer records the path of each message through an integration flow, including each hop, transformation, and outcome. For IRIS-managed resources, the message trace, workflow state,

data changes, and audit log are held in one platform, so the business side of an end-to-end account does not have to be stitched together across products. A correlation identifier carried from the agent's request lets that record be joined with agent traces and with connected application records.

Model traces explain reasoning and orchestration behavior. IRIS audit and workflow records establish what occurred within the entire platform's scope. Connected application records establish what those systems accepted and committed. Together, these records support an end-to-end account without assuming that any single trace is authoritative for the entire distributed process.

Agents may propose and request actions. The platforms and applications responsible for those actions determine whether they are permitted and record what actually happens.

8. Implementation Patterns

The following patterns of increasing complexity illustrate how these architectural principles can be applied.

Pattern 1: Operational RAG with Native Vector Search

This pattern grounds an agent in both semantic content and current operational facts. Source text is segmented and converted into embeddings. Vectors are stored with identifiers, metadata, timestamps, security attributes, and links to the records they describe.

The platform automatically constrains the searchable corpus according to the task and authenticated user's permissions at query time. Vector search identifies semantically relevant candidates. SQL predicates and application logic filter by customer, case, status, or other operational attributes. Object, document, or key-value access can supply additional current state.

The model receives a bounded context package with provenance. If the agent subsequently proposes an action, the retrieval result does not grant execution authority. The action is submitted through a separately governed tool or workflow, which revalidates it against current state. For IRIS-managed resources, that tool or workflow is an IRIS capability; for connected applications, the application performs the check.

This pattern is appropriate when semantic relevance must be combined with current operational attributes. It is less suitable when the task can be satisfied through a precise deterministic query or when the source content cannot be governed at the required granularity.

Pattern 2: An External Agent Invokes a Governed Enterprise Action

In this pattern, an agent built in an external framework uses IRIS and a supported Model Context Protocol connection to discover and request capabilities exposed from IRIS. A capability might retrieve an order, validate eligibility, calculate an adjustment, create a case, or start a business process.

The capability contract defines accepted parameters and returned values. Existing IRIS business logic, such as a method, query, or workflow, can be

22

published as a capability without separate integration code, so the logic an organization has already built and tested is what the agent invokes. The implementation remains within the governed platform. The agent can select and request the capability, but it does not receive direct database access or arbitrary code-execution rights.

When the request arrives, IRIS authenticates the connection, establishes the configured identity, validates the operation and inputs, and authorizes access to IRIS-managed resources. Business logic checks current record state. If approval is required, the request becomes a pending workflow item rather than an immediate write. Calls to external systems use approved interfaces and remain subject to those systems' controls.

The result is bounded and the execution record associates the requesting agent, operating identity, selected capability, validation, policy decision, workflow or transaction outcome, and resulting changes.

Pattern 3: Agent Participation in a Real-Time Integration Workflow

This pattern applies when work begins with an event and spans several systems and participants. An integration flow receives an event through an adapter, API, message queue, file, or industry protocol; validates and transforms the payload; enriches it with operational data; and starts a durable process.

The process invokes an agent only for steps that benefit from interpretation, synthesis, or dynamic planning. The agent may summarize the event, compare it with similar cases, identify missing information, or propose a response. Deterministic workflow logic continues to manage sequencing, service calls, timers, retries, and state transitions, because for structured steps like these, deterministic code is typically more predictable and cost-efficient than probabilistic methods. High-impact decisions and required approvals are routed to a human task with supporting context.

After approval or deterministic validation, IRIS invokes the downstream services and records the resulting operational updates. Correlation identifiers link the inbound event, agent invocation, retrieved context, workflow instance, human decision, outbound messages, and transaction results, so the full path can be traced afterward. Because process state is durable, processing can resume after an interruption without asking the model to reconstruct what happened.

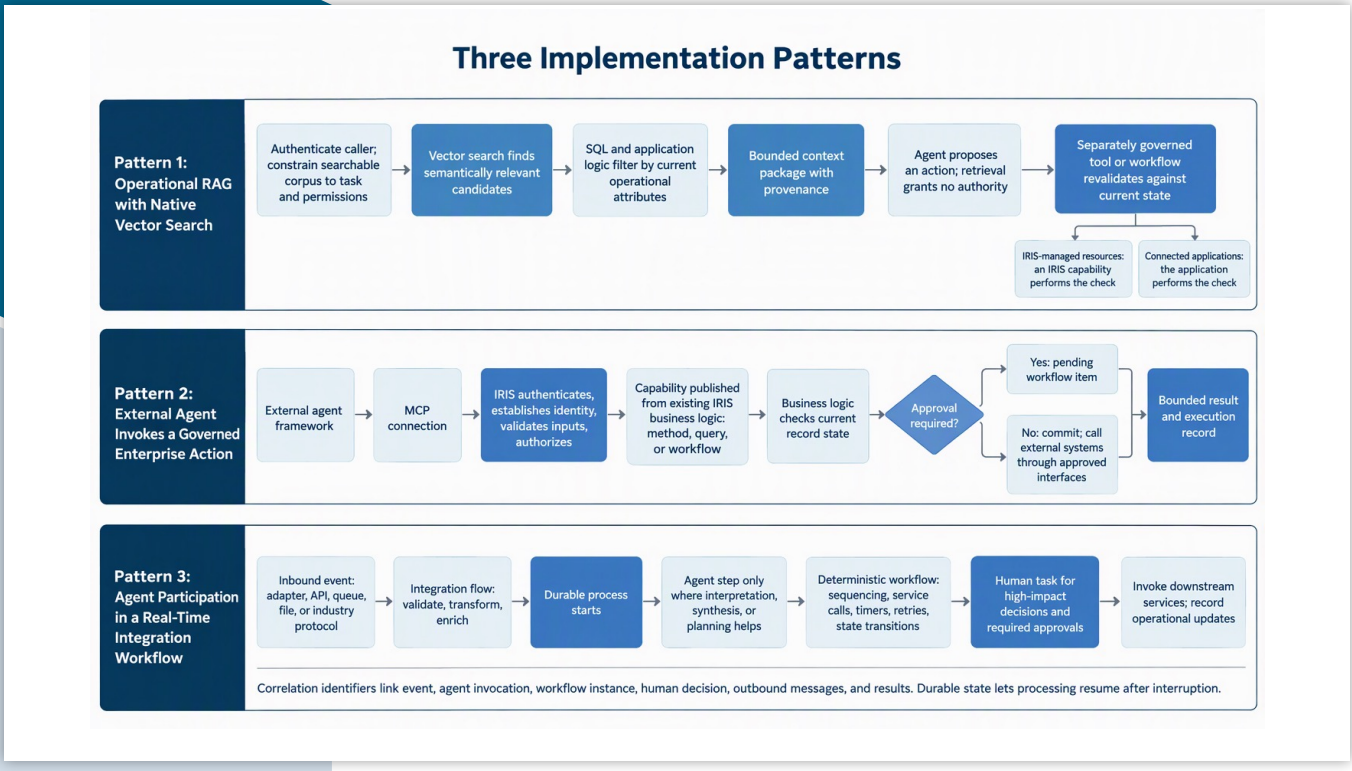


Figure 5. The three implementation patterns: operational RAG, an external agent invoking a governed action, and agent participation in an integration workflow.

9. Reference Architecture

A production reference architecture begins with channels through which work enters the system: applications, users, APIs, events, scheduled processes, devices, or external agents. Models and agent frameworks interpret objectives, manage reasoning, and select approved capabilities. They are not the authoritative source of enterprise data, permissions, transaction state, or process completion.

Layer 1: Channels and Events

Requests originate from users, applications, event streams, devices, partners, scheduled jobs, and external agents. Each entry point should establish source, identity, provenance, and correlation information appropriate to the process.

Layer 2: Models and Agent Frameworks

Models provide interpretation and generation. Agent frameworks manage session state, planning, model selection, retrieval, tool discovery, and agent-to-agent coordination. Guardrails at this layer address prompt handling, model access, output structure, and behavioral constraints.

Layer 3: InterSystems IRIS and Governed Interfaces

InterSystems IRIS and platform interfaces present bounded data, methods, services, queries, and workflows. They authenticate requests, establish operating identities, shape outputs, and route requests into IRIS-managed capabilities. External agents can connect through supported interfaces such as MCP; native agents can operate close to the operational environment.

Layer 4: Operational Data and Context

The IRIS operational data plane provides current records, transactions, documents, objects, key-value structures, vectors, analytical results, and application state. IRIS Interoperability contributes recent events. IRIS workflow services contribute process state. Identity services contribute current entitlements. Business logic contributes rules and constraints. Together, these elements form task-specific operational context.

Layer 5: Workflow and Interoperability

Integration flows built with IRIS Interoperability receive and route messages, transform formats, invoke business rules, call downstream services, and maintain process visibility. Durable workflows manage deterministic steps, long-running state, retries, timeouts, approvals, compensation, exceptions, and recovery.

Layer 6: Systems of Responsibility

Enterprise applications, systems of record, identity platforms, partner systems, and external services remain authoritative for the resources they own, enforcing their own controls and recording what they commit.

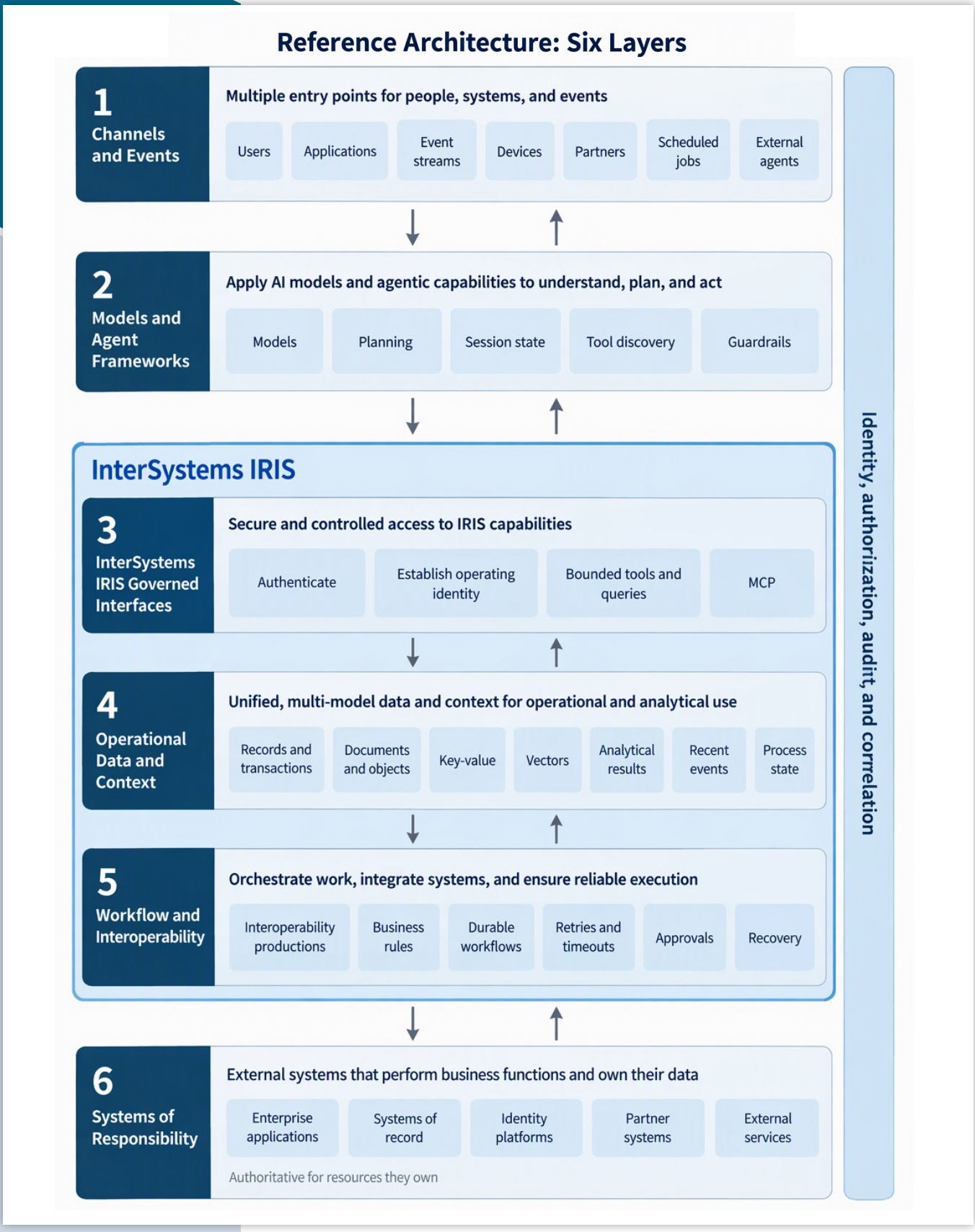


Figure 6. The six layers of the reference architecture. Model reasoning stays flexible while authority, state, and execution controls remain with the systems responsible for the work.

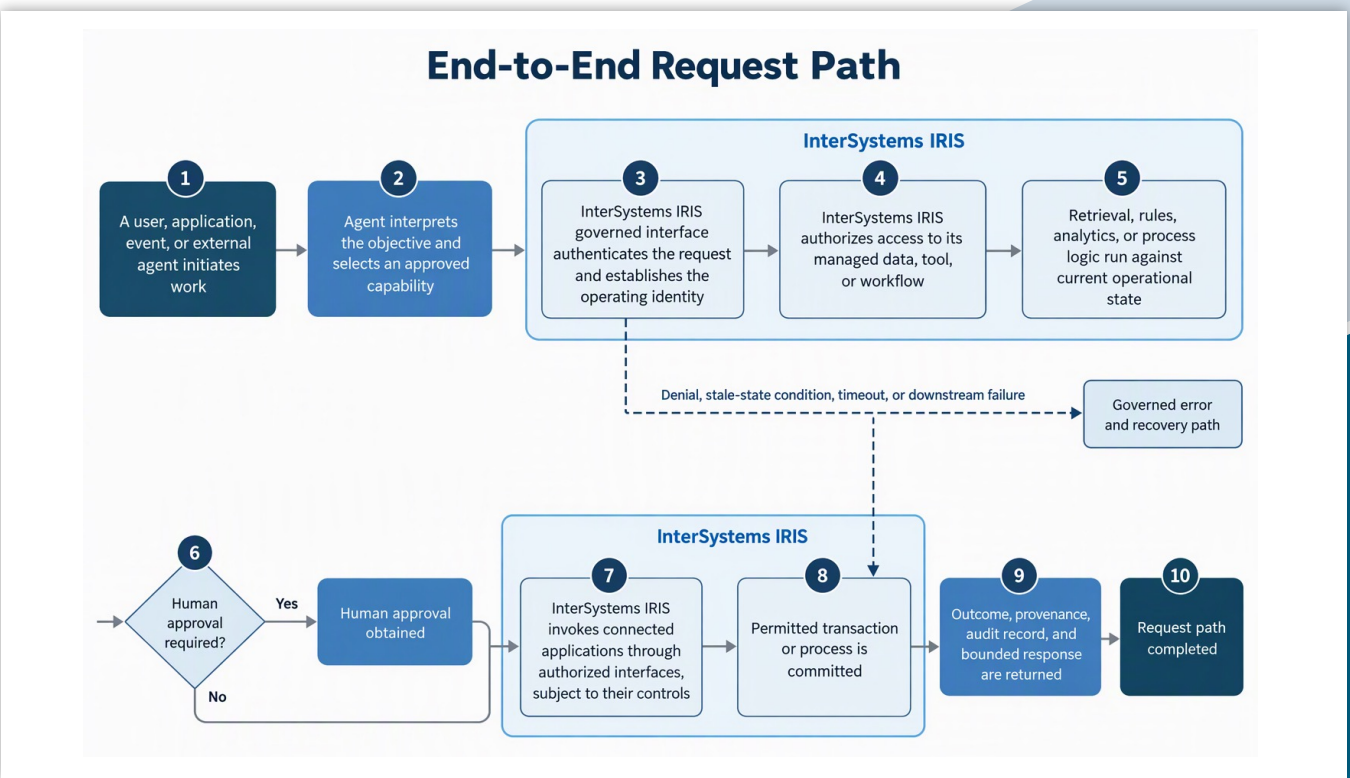


Figure 7. The end-to-end request path, from initiation through governed execution to a recorded outcome, with a governed recovery path for failures.

1. A user, application, event, or external agent initiates work.
2. The agent interprets the objective and selects an approved capability.
3. IRIS or another governed interface authenticates the request and establishes the operating identity.
4. IRIS authorizes access to its managed data, tool, or workflow.
5. Retrieval, rules, analytics, or process logic execute against current operational state.
6. A human approval is obtained when required.
7. IRIS invokes connected applications through authorized interfaces, subject to their controls.
8. The permitted transaction or process is committed.
9. The outcome, provenance, audit record, and bounded response are returned.
10. A denial, stale-state condition, timeout, or downstream failure follows a governed error and recovery path.

Questions to Test Any Agent Architecture

These questions apply to any architecture, whatever products it uses. They help distinguish a demonstration that works from a system that can be trusted with consequential work.

- Where does the authoritative state of each business process live, and what happens to it if the model, agent framework, or runtime is replaced?
- At execution time, which system revalidates the action against current state, and how many handoffs separate it from the data?
- Which identity does each step run under, and where is that enforced?
- What happens after a timeout, duplicate request, or partial completion?
- Can a single correlated record show the request, authorization, approval, and committed outcome?
- Do agents see bounded business capabilities, or raw access to databases and APIs?
- How does a human approval pause and resume a long-running process?
- Which parts of the design depend on a particular model, framework, cloud, or application suite?

Conclusion: Connecting Intelligence to Execution

The central challenge of enterprise AI is no longer generating accurate responses. For organizations moving agents into operational processes, it is letting intelligent systems do real work under the same controls as the rest of the business.

Success requires more than better models and broader tool access. Agents must work from current, authoritative state, act through bounded capabilities, and operate within processes that can recover from failure and show what happened.

InterSystems IRIS brings these elements together in one operational platform and extends them to native and external agents, alongside the AI and application investments organizations already have.

The result is a real-time data platform for agentic AI: an operational foundation through which agents can use current enterprise context, participate in durable workflows, invoke governed capabilities, and contribute to real business outcomes without becoming the source of authority for the systems they touch.

As agents take on mission-critical work, the deciding question for architects shifts from which model to use to how safely agents can be connected to enterprise execution. The operational foundation described in this paper is designed to answer that question.

InterSystems IRIS connects agents to real-time operational context, enterprise workflows, and governed actions so they can participate safely in mission-critical operations.

For more information, visit www.intersystems.com/aihub

InterSystems®
Creative data technology

© 2026 InterSystems Corporation. All rights reserved.
For trademark information, see InterSystems Notices, Terms & Conditions. 102926