RJL SOLUTIONS

ARCHITECTURE PAPER / CONNECTED ENTERPRISE

The Four-Part System

An enterprise data and AI architecture, organized around four responsibilities and the contracts that connect them.

BRAIN · NERVOUS SYSTEM · BODY · IMMUNE SYSTEM

Richard White
RJL Solutions · rjlsol.com

OCTOBER 2026

01 / THE PREMISE

Four responsibilities, delivered as one capability.

Enterprise AI needs a dependable route from business intent to information, permitted execution and an observable result. The Four-Part System organizes that route around four responsibilities. Deliver them together for a bounded workflow, and a useful answer also comes with agreed meaning, controlled authority and continuing ownership.

This paper is for data, technology, AI and security leaders, enterprise architects and domain product owners. It explains component boundaries and the contracts that connect them. It also develops the Brain and Operating Layer reference architecture into a vendor-neutral design a team can review against its existing services.

PART	ARCHITECTURAL RESPONSIBILITY
Brain	Approved business entities, relationships, metrics, knowledge context, and answer routing
Nervous system	Versioned APIs, tool interfaces, events and commands that carry approved work
Body	Delivery and operation: environments, provisioning, workflows, deployment, support, and product lifecycle
Immune system	Classification and exposure intelligence, coordinated policy, enforcement, and security response

The parts describe responsibilities. They aren't four products to buy or four teams to fund separately. A control plane, described in section 08, connects the four without adding a fifth. One platform may implement several responsibilities, and one responsibility may span several platforms. The design holds together when each interface has a contract, each decision has an owner and you can test the actual request path.

How to read it

The system view and the brain's layers come first. The sections after that establish information contracts, execution, environments, security and control evidence, then follow an illustrative claims request through all four parts. The closing sections cover change, organization, implementation and review.

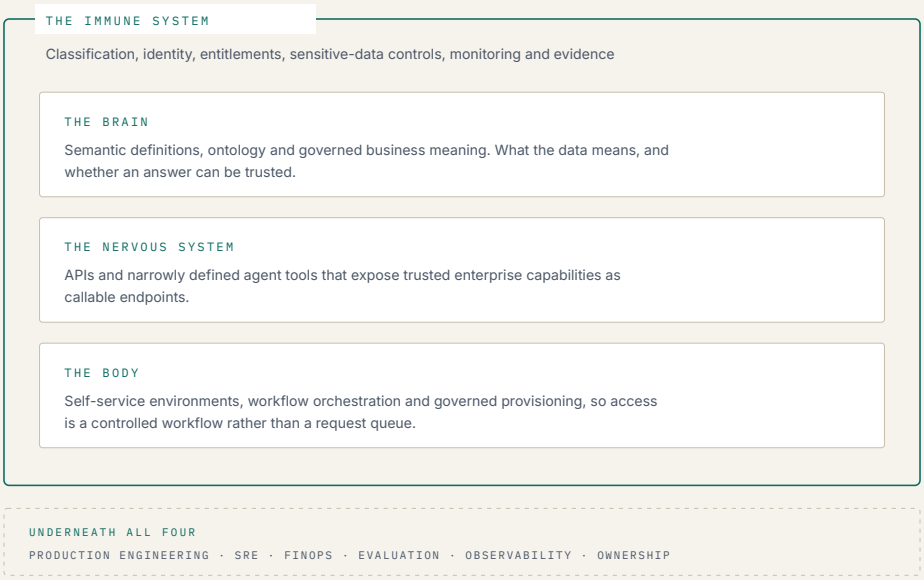
In this reference architecture, the Operating Layer is the operating system around the Brain: the provisioning and operating pattern that makes it usable. It's a pattern. You can't buy it. The architecture can use existing infrastructure wherever its contracts and controls meet the selected workflow's needs.

THE STARTING POINT — One complete governed capability. Extend its users, sources or authority after the business results and the control evidence support the next scope. The architecture becomes reusable through maintained contracts and working services, rather than through the diagram alone.

02 / THE SYSTEM VIEW

Read the connections as contracts.

The system view shows logical responsibilities and how they interact. It isn't a network topology or a mandatory call sequence. Security enforcement can appear at several relevant interfaces. And provisioning normally happens before the user's request, not inside every request.



Four responsibilities in one system, resting on production engineering disciplines. The immune system is drawn as a frame rather than a box, because it governs every path into the other three.

The brain picks the approved meaning and information route for a question. The nervous system exposes the operations available on that route. The body supplies the environments, identities, deployment workflow and service arrangements needed to run it. The immune system supplies security context and keeps tested enforcement in place across the participating systems.

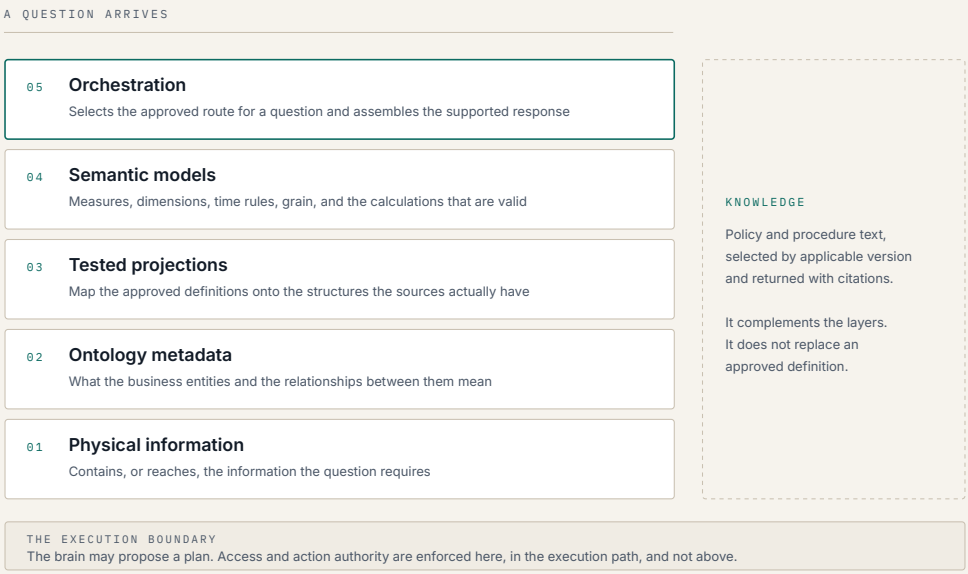
Existing operational systems, analytical stores and document repositories support the capability through governed products and interfaces. The four-part view doesn't displace them. The architecture should explain which source is authoritative, and how the consumer reaches it under the applicable controls.

The control plane connects identities, product and policy versions, approvals, release decisions and evidence. Its services can be distributed. Their authoritative records and synchronization rules have to be explicit, so an operator can explain both the result and the conditions that permitted it.

THE REVIEW TEST Whether the team can trace one real request through its implementations, and explain what happens when a required dependency or control fails.

The brain gives information business meaning.

The brain connects a question to approved entities, relationships, calculations and knowledge. It should expose ambiguity. Filling a missing business decision with a plausible interpretation is the wrong move. Its five conceptual layers describe responsibilities. They don't require five separately purchased products.



The five layers, read from the question down to the source. Knowledge retrieval sits alongside them rather than inside them, and authority is enforced below all of it.

Generated, tested views over the sources (dbt models, for example) are one implementation option. Validate source mappings and known results before deployment. A successful compilation establishes structural compatibility. It says nothing about business correctness. Relationship or graph capabilities help where the workflow needs traversal or hierarchy reasoning. A metric question can use an approved semantic interface without any graph.

WHO APPROVES MEANING

A customer can be a policyholder, an insured person, an employer or another role depending on the task. Govern the valid differences and their intended contexts. Domain owners approve definitions and exceptions; engineering encodes and tests them. Preserve version and effective-date context when a definition changes an answer.

TYPICAL COMPONENTS

Categories, not products. A governed data platform, a transformation layer with tested models, a semantic layer, a catalog and glossary, and a knowledge index that keeps source-level permissions. Any combination that preserves the contracts will do.

04 / INFORMATION

Information products carry consumer commitments.

The four parts depend on information that stays trustworthy after its original project ends. A data product is an owned, versioned unit of information or capability, with an approved interface and a consumer contract. A knowledge corpus needs equivalent commitments for authority, applicability, permissions and upkeep.

CONTRACT AREA	COMMITMENT TO DECLARE AND DEMONSTRATE
Meaning	Purpose, grain, identifiers, definitions, relationships, and known limitations
Interface	Supported schema, operations, version compatibility, and change notice
Information state	Quality, freshness, source provenance, and behavior when requirements fail
Permitted use	Classification, entitled consumers, purpose restrictions, and enforcement path
Continuing operation	Owner, support, service objectives, cost limits, and retirement process

A foundational product is owned close to the domain that produces the information. A derived product owns the extra joins, reshaping, calculations and consumer commitments it introduces. Reusing a certified source doesn't automatically certify new logic built on top of it.

Maintain knowledge through its derived artifacts

Parsing and indexing create new ways to reach information. Keep the source identifier, version, effective date, applicable business context and restrictions intact through chunks, indexes and the relevant caches. Apply entitlement before restricted content enters the model's context.

Define how replacement, deletion and permission revocation reach those artifacts. Set an update commitment, plus a restriction or disclosure for when the corpus is known to be stale. A relevant paragraph isn't enough evidence that it's current, or that it applies to the case.

Connect certification to current behavior

Store declarations in versioned artifacts, and use them in validation, publication, deployment and monitoring. A manifest describes the commitment. Tests and operational evidence demonstrate it. Name who responds when the source or the derived product stops meeting the contract.

For the first workflow, certify the dependencies it needs. Keep changing source state, or as-of time, visible where it matters. Maintain compatibility rules for sources that evolve independently of the consuming agent or application. Catalog discovery should show the approved scope and current condition. It shouldn't imply unrestricted use.

05 / THE NERVOUS SYSTEM

The nervous system carries approved operations.

The nervous system connects consumers to governed capabilities through interfaces with explicit contracts. It carries identity and request context, invokes bounded operations and returns observable outcomes. The protocol you pick doesn't decide what a caller may do, or what the information means.

INTERFACE	CONTRACT AND FAILURE BEHAVIOR TO ESTABLISH
Read	Meaning, query scope, freshness, result schema, pagination, and access enforcement
Command	Business operation, validation, authorization, approval, duplicate protection, durable state
Event	Producer authority, schema, event identity, ordering assumptions, replay, consumer recovery
Agent tool	Named operation, typed inputs and outputs, approved dependencies, limits, and side effects

These contracts can share infrastructure and still be reviewed independently. Reads may tolerate a declared snapshot. Commands need authoritative outcomes. Event consumers tell repeated delivery from new work. Tools need a scope narrow enough to validate the proposed action.

Preserve identity to the point of execution

For delegated requests, identify the initiating person or application and the agent acting for them. Three things bound effective authority: the caller's entitlement, the tool's approved scope, and purpose or context restrictions. Background work needs an explicitly approved workload identity and scope.

Validate the target and typed arguments before execution. Carry verified authorization context through gateways, services, sources, pooled connections, asynchronous jobs and caches, as a token scoped to each downstream hop. Never forward the caller's credential. An identity check at ingress doesn't establish that a downstream read or write respects the same boundary.

Expose the operation the business permits

Prefer a narrow operation such as `get_claim_context` to arbitrary query or update access, when the task can be served that way. Validate the returned schema and failure state before treating the work as complete. A callable tool, including one exposed through the Model Context Protocol (MCP), still needs the contract and the enforcement.

For writes, define accepted, applied, rejected and unresolved states as the business operation needs. Retry an uncertain request without reconciling its durable state and you can duplicate an effect. An idempotency mechanism reduces duplicate execution within its defined scope. It can't prove the action was right for the business.

TYPICAL COMPONENTS Categories, not products. An API gateway, an operation registry with typed contracts, an authorization service, and a query layer over certified products.

06 / THE BODY

The body makes the capability usable and supportable.

The body provisions environments, deploys approved artifacts, manages access workflows and connects delivery to service operation. Rapid provisioning helps when the resulting environment also has ownership, bounded access and a path to support.

LIFECYCLE STATE	OPERATING COMMITMENT
`sandbox`	Approved experimental data limits, ownership, and clear user expectations
`managed_sandbox`	Controlled real data, named dependencies, and tested request paths
`certified`	Accepted scope, enforced contracts, support, monitoring, and release evidence
`degraded`	A recorded breach or uncertainty restricts use through enforced conditions
`deprecated`	Replacement and migration deadline given; agreed service continues
`retired`	Consumers migrated; access, endpoints and retained artifacts addressed

Promotion changes the obligations. An endpoint that works is no reason to promote on its own. Before relying on an environment, a consumer should be able to see its status, approved users, available operations and support route.

Provision from approved declarations

Use versioned templates and deployment artifacts for the environment, identities, configuration, tools and observability. Resolve required references, and validate the deployed state against the reviewed release. Access workflows should identify the decision holder and record the approved scope, conditions and, where it applies, expiry.

Keep credentials in the approved identity or secret system. A deployment pipeline may carry out an approval. It shouldn't grant itself the business or control authority to make that decision. Detect and reconcile configuration drift that changes the reviewed capability.

Build operation into delivery

Assign service objectives, limits, incident ownership, rollback and a recovery exercise. Bring internal engineers into implementation and support from the start. Before accepting transfer, have the receiving team show it can investigate a request, restrict a route and run a change.

The body also connects the capability to the business process: adoption, user guidance, exception handling and outcome measurement. An interface that creates more verification work can hit its technical service objective and still fail the business task.

TYPICAL COMPONENTS	Categories, not products. An orchestration and workflow engine, release pipelines with gates, observability and tracing, and incident and change management.

The immune system sustains security across routes.

The immune system discovers exposure, supplies security context, coordinates policy and keeps enforcement and response working across the capability. Identity, source platforms, gateways, tools, monitoring and security operations share these responsibilities. Classification alone enforces no access decision.

RESPONSIBILITY	EVIDENCE THAT THE IMPLEMENTATION PERFORMS IT
Discover and classify	Sources and derived artifacts have maintained sensitivity and exposure context
Decide permitted use	Authoritative identity, purpose, classification and policy determine the allowed scope
Enforce the decision	Relevant execution points filter, mask, deny, or constrain the operation
Detect and respond	Control failures reach named responders who can contain the affected path

Record the policy authority and the enforcement point for each consumption route. NIST's Zero Trust Architecture distinguishes policy decision from policy enforcement.[1] The enterprise implementation needs a tested connection between those responsibilities, including for when the decision changes or cannot be established.

Test the routes that bypass assumptions

Test a direct API call, an agent tool, retrieval, an application and the relevant cached or asynchronous work. Confirm the selected controls apply on every permitted path. Never treat a semantic model or a catalog description as the only access boundary for the information it describes.

For policy propagation, declare the authoritative record, the allowed delay, the reconciliation and the failure response. An uncertain entitlement or tenant boundary should deny the affected request, or invoke an explicitly approved restricted mode. When a control fails, don't quietly carry on with a broader credential or a different source.

Keep untrusted content from granting authority

A retrieved passage or a tool response can contain instructions that try to redirect an agent. Preserve the distinction between task evidence and operating instructions. OWASP recommends enforcing tool authorization in the execution component, outside the agent's context.[2] Test hostile content alongside legitimate tasks: a prompt instruction alone does not establish the boundary.

Security telemetry and request evidence need their own access and retention controls. Don't create a second uncontrolled store of sensitive prompts, results or credentials in the name of observability.

TYPICAL COMPONENTS	Categories, not products. An identity provider with workload identity, a policy decision point, data classification and entitlement tooling, and secrets management.

Connect decisions and evidence.

The data plane carries reads, retrieval, events, commands and results. The control plane coordinates the context that governs those operations: identities, contracts, classifications, policies, approvals, versions, limits and evidence. It connects the four responsibilities. It isn't an extra business capability.

Define authoritative records and enforcement targets

A shared control plane can use distributed services. Name the authority for each decision, its enforcement targets and the response when they disagree or synchronization fails. Define each service's reliability and dependency commitments.

Version the artifacts that affect behavior, and link approval and evaluation to the deployed release. Track source state and effective dates. A release identifier can't explain a source or procedure that changed on its own.

Retain the trace an authorized reviewer needs

Connect the caller and workload identities, the selected definition, the policy outcome, source access, tool calls, component versions, response or command state and the observed outcome under one correlation identifier. Keep references, redaction and snapshots as the workflow's evidence requirements demand.

Reconstruction supports investigation. It doesn't promise identical regeneration of a probabilistic answer. Define the necessary evidence, the authorized reviewers, the retention and how to retrieve it after a service failure or a release change.

Use the same discipline at release and at runtime

Deployment checks validate contracts, references, evaluations and controls. Monitoring exposes breaches. When a source or a policy no longer supports approved behavior, restrict the affected path and bring in its decision holders.

REPORT WHAT YOU CAN SHOW	The service condition the team can demonstrate now, rather than the one the design intended.
TYPICAL COMPONENTS	Categories, not products. A registry of products and versions, an evidence store with immutable references, an approvals workflow, and a request-trace store.

09 / WORKED EXAMPLE

Follow an illustrative claims request.

An examiner asks: “Who is connected to this claim, what is its current status, and which procedure applies?” The first capability supports information gathering. The examiner keeps the case decision. Adjudication, payment authorization, record updates and external communication stay outside its scope. This example illustrates the architecture. It reports no client result.

Establish authority and select the route

Authenticate the examiner, identify the agent release, and evaluate the permitted claim population and purpose. The brain resolves the task and the context it needs. An ambiguous claim identifier or product context should trigger clarification. An assumed match is the wrong answer.

The nervous system exposes approved read tools for case context and applicable procedure. The immune system’s controls are enforced where those tools reach their sources and return information. Before the request begins, the body’s environment and deployment workflow establish the approved configuration and the support route.

Assemble information with explicit provenance

Use approved relationship rules to tell the insured person, the policyholder, the policy and the employer apart where they apply. Read current status from the designated operational interface, or from a product that meets the required freshness. Retrieve a procedure using product, jurisdiction where relevant, effective date and access restrictions.

The answer separates source facts from interpretation, flags unresolved relationships and cites authorized supporting material. If a later request asks for a benchmark, use the approved metric definition and disclose its period. If required evidence is missing, return the defined incomplete state or escalate.

CONDITION	BEHAVIOR AND EVIDENCE TO VERIFY
A revoked examiner repeats the request	Denial holds through tools, sources, retrieval and the relevant caches
Status or procedure is stale	Freshness is disclosed, or the affected task is restricted as agreed
A passage asks for a payment action	Write and external communication tools remain unavailable
Two relationships conflict	The answer identifies the uncertainty and routes it to review

Keep the request trace with its identities, definitions, source context, policy decisions, versions and outcome. Measure the time to assemble usable context against the baseline, verification and exceptions included.

Change and failure are architectural behavior.

Contracts stay useful when the system can explain how a change affects consumers, and what happens when a dependency fails. A failure response should keep the approved authority and disclose incomplete work. Returning a plausible answer through an unapproved route hides exactly the condition the architecture needs to manage.

CHANGE	ASSESSMENT AND EVIDENCE BEFORE RELEASE
Definition or source mapping	Affected consumers, known answers, compatibility, and historical context
Prompt, model or tool	Scope, routing, answer quality, refusals, and execution boundaries
Policy, corpus or permission	Applicability, entitlement, propagation, retrieval, and cache behavior
Command or event behavior	Durable state, replay or retries, duplicate effects, and consumer recovery

An automated compile step can catch a missing column or an incompatible schema. Business validation still has to test whether the resulting relationships and calculations mean what the domain approved. A release should identify the exact artifacts and the scope its evidence supports.

Recover without widening authority

Declare useful-completion, availability, latency and freshness objectives where relevant, with measurement windows and response conditions. Define safe behavior for unavailable identity or policy, stale sources, failed tools, exhausted limits and broken compatibility. Name who can disable a route, restrict a population or roll back an artifact.

> **A last approved version is a recovery option only if it still meets current policy and dependency requirements.**

Restoring older software doesn’t restore a revoked entitlement. It doesn’t make a superseded procedure applicable either. Test recovery against the commitments the workflow needs now.

Treat business effects separately from software rollback.

For consequential actions, validate the command and any required approval at execution. Keep enough state to tell acceptance from application, and reconcile uncertain outcomes before retrying. Use duplicate protection within a defined operation and retention scope.

Where an asynchronous command and an event have to stay consistent, choose a durable coordination pattern and test how it recovers. Replayed events or repeated requests shouldn't create unintended duplicate business effects.

A software rollback may leave a completed transaction in place. Remediation needs its own authorized business process. Recovery exercises should demonstrate the team, the runbook and the authority to use it. Restore the capability once the cause is fixed and the relevant regression evidence supports the proposed scope.

Organize around decisions and service commitments

The four-part architecture needs joint delivery across the business, data, technology, AI and security functions. It doesn't need four new departments. It needs named decision holders, and a team able to resolve the dependencies of one complete capability.

RESPONSIBILITY	DECISION OR COMMITMENT TO OWN
Business process	Workflow changes, outcome measure, adoption, and acceptance
Domain meaning and products	Definitions, source commitments, derived logic, and consumer change notice
Platform and delivery	Environments, integration, deployment, and reusable services
Security and controls	Permitted use, enforcement conditions, review, and incident coordination
Capability and service	Scope, evaluation, lifecycle, support, recovery, and recurring cost

A product owner coordinates the capability without inheriting the control authority of every participating team. Record who can approve a definition, grant access, accept residual risk, stop a release and fund maintenance. In a small team, one person can hold several roles, as long as those responsibilities stay explicit.

Balance shared services and domain ownership.

Shared teams maintain reusable identity, contract standards, deployment, monitoring, evaluation tooling and catalog infrastructure, wherever those services fit the task. Domain teams keep business meaning, workflow authority and source responsibility. Agree the service commitments between them, and an escalation route for conflicts or unmet dependencies.

Set a review cadence that suits criticality and the rate of change. During early operation, review incidents and degraded dependencies often. Review adoption, exceptions and outcome evidence with the business owner. Before material releases, examine the changed scope and the affected evaluation with the relevant decision holders.

Prove transfer through operation

Bring internal engineers and service owners into implementation, tests and support from the start. Have the receiving team demonstrate user admission and revocation, request investigation, change assessment, restriction and recovery in the agreed environment.

The evidence should show the team has the access, the authority, the skills and the funded capacity to carry its responsibilities. A repository, a runbook or a training session alone doesn't establish that. Before extending the service or completing transfer, record the remaining gaps with owners and acceptance conditions.

Implement one complete governed capability

Pick a repeated task with a measurable outcome, a business owner, identifiable users and reachable dependencies. Keep the first scope narrow enough to test the full request path. Reuse mature services where they meet the required contracts. Build the gaps that decide whether the workflow can run.

PART	EVIDENCE THE FIRST CAPABILITY SHOULD PRODUCE
Brain	Approved entities and definitions, correct route, known answers, and source context
Nervous system	Narrow versioned interfaces, complete identity path, bounded operations, failure states
Body	Controlled environment, reviewed deployment, limits, support, receiving-team demonstration
Immune system	Maintained classification and policy, enforcement tests, revocation, and incident route
Control plane	Linked versions, approvals, request traces, release evidence, and current condition

Sequence decisions and engineering together.

Implement the architecture for the stated task and users, with complete evidence for their scope. Extend definitions, sources and platform services as later workflows need them.

Frame the workflow, baseline, ownership, exclusions and deciding dependencies first. Prove meaning, authority, usefulness and failure behavior with representative requests in the approved environment. Then build the accepted slice into a supportable release, and have internal owners run it.

Review the outcome and control evidence before extending users, sources or actions. Proceed when it supports the next scope. Reshape when dependencies or assumptions need to change. Stop when value, or a workable operating path, no longer justifies the investment. The 90-day first-capability pattern can organize this work. The actual schedule depends on access, decisions and integration constraints.

Record component choices as bounded decisions

For each component, document the responsibility it implements, its authoritative scope, its interfaces, its owner, its dependency risks and its replacement conditions. Validate the required maturity, availability, deployment location, licensing, performance and enforcement behavior for the actual implementation.

Choose components by the responsibilities they carry and the evidence they have to supply. A replacement is compatible when it keeps the relevant contracts and control behavior, with migration and evaluation accounted for.

> **A similar interface alone does not demonstrate that equivalence.**

13 / THE REVIEW

Review the architecture against a real request.

Bring one proposed workflow, a request trace or design walkthrough, and the current dependency and owner map. The job is to find which commitments have evidence, which are still assumptions and what decision you can make for the named scope.

01 Can the business owner name the useful outcome, and change the workflow that produces it?

02 Can the team identify the approved definitions, sources, applicable knowledge, and their current consumer commitments?

03 Does each operation have a bounded interface, an enforcement point, and a tested identity and authority path?

04 Can a reviewer reconstruct the request, policy decisions, source context, versions and outcome under the required evidence controls?

05 Do failed dependencies and exhausted limits produce an agreed safe response, and can the receiving team demonstrate recovery or restriction?

06 Are the release decision, continuing ownership, maintenance funding and conditions for expansion explicit?

Decide the next bounded action

Give each gap that matters an owner, a closure criterion and a target decision date. Choose the evidence that would demonstrate closure. A finished design document proves nothing about runtime behavior. A critical failure in access, isolation, unauthorized action, integrity or required evidence blocks the affected release, whatever the average answer quality.

Use the **Connected Enterprise Data & AI Blueprint** for the wider business and delivery context, **Agent Onboarding** for the agent and tool contracts, and the **10-Question Diagnostic** for the working session. Bring the workflow, its request path and its open decisions to **rjlsol.com/contact**.

[1] NIST SP 800-207, Zero Trust Architecture, August 2020. Policy decision and enforcement responsibilities.

[2] OWASP Cheat Sheet Series, AI Agent Security Cheat Sheet, accessed October 2026. Authorization outside the agent context, and testing of agent boundaries.

The diagrams express logical responsibilities. Implementation evidence and the organization’s decision holders determine approval for the actual scope.

ABOUT

Richard White

I was global firm chief data officer at McKinsey & Company, owning data, AI and platforms end to end, and before that chief data officer at New York Life, where I built the chief data office and reported data risk to risk committees at multiple levels. Earlier I led data at the Federal Reserve Bank of New York and enterprise data governance at Citigroup. I founded RJL Solutions in 2024. My book, *The Day After the Demo*, is free at rjlsol.com/book.

RJL Solutions connects strategy, architecture, engineering, governance and operation through Frame, Prove, Build, and Run & Transfer. Each phase is separately scoped and priced in a written proposal before you commit. The first sixty-minute working session is free.

GUIDE **The Connected Enterprise Data & AI Blueprint**
The business and delivery context around this architecture, in 24 pages.

TECHNICAL **Agent Onboarding: Treating an Agent Like a Data Product**
The agent and tool contracts this architecture assumes, in 13 pages.

WORKSHEET **The 10-Question AI Production-Readiness Diagnostic**
Four pages to complete with your business owner, technical lead and control owner.

RJL SOLUTIONS

WEB rjlsol.com

EMAIL rwhite@rjlsol.com

This paper describes no client engagement and no client result. The claims workflow is illustrative. Before production adoption, any implementation has to validate platform maturity, regional availability, licensing, regulatory obligations, performance and operational ownership.