RJL SOLUTIONS

ARCHITECTURE AND OPERATION

The Federated Data Platform

A practical architecture for domain delivery, shared services and consistent enterprise control.

CLEAR OWNERSHIP · REPEATABLE CONTROLS · OPERATING EVIDENCE

Richard White
RJL Solutions · rjlsol.com

ARCHITECTURE UPDATE
29 OCTOBER 2026

01 / THE EXECUTIVE CASE

Build reuse into the next use case.

An enterprise platform earns its value when each useful delivery leaves a supported capability behind.

A large foundation program can postpone business use while someone builds the platform. Independent use-case projects can deliver quickly, but they repeat integrations, definitions and controls. A product-led approach looks for an incremental alternative. Deliver a bounded outcome, and build contracts and services the next consumers can reuse as you go.

Reuse can cut repeated work. It isn't free. Product ownership, documentation, compatibility, support and shared services all create ongoing costs. Weigh those costs against observed consumption and avoided duplication. A reuse curve on a slide proves no savings.

LEADERSHIP DECISION	PRACTICAL TEST
What remains domain-owned?	Meaning, products, pipelines and the business outcome have accountable owners.
What is provided once?	Common services solve repeated needs and carry explicit commitments.
Where are controls applied?	Every storage and serving path has tested enforcement; exceptions are recorded.
How does the next use improve?	A later team can reuse a product or template without bespoke reconstruction.

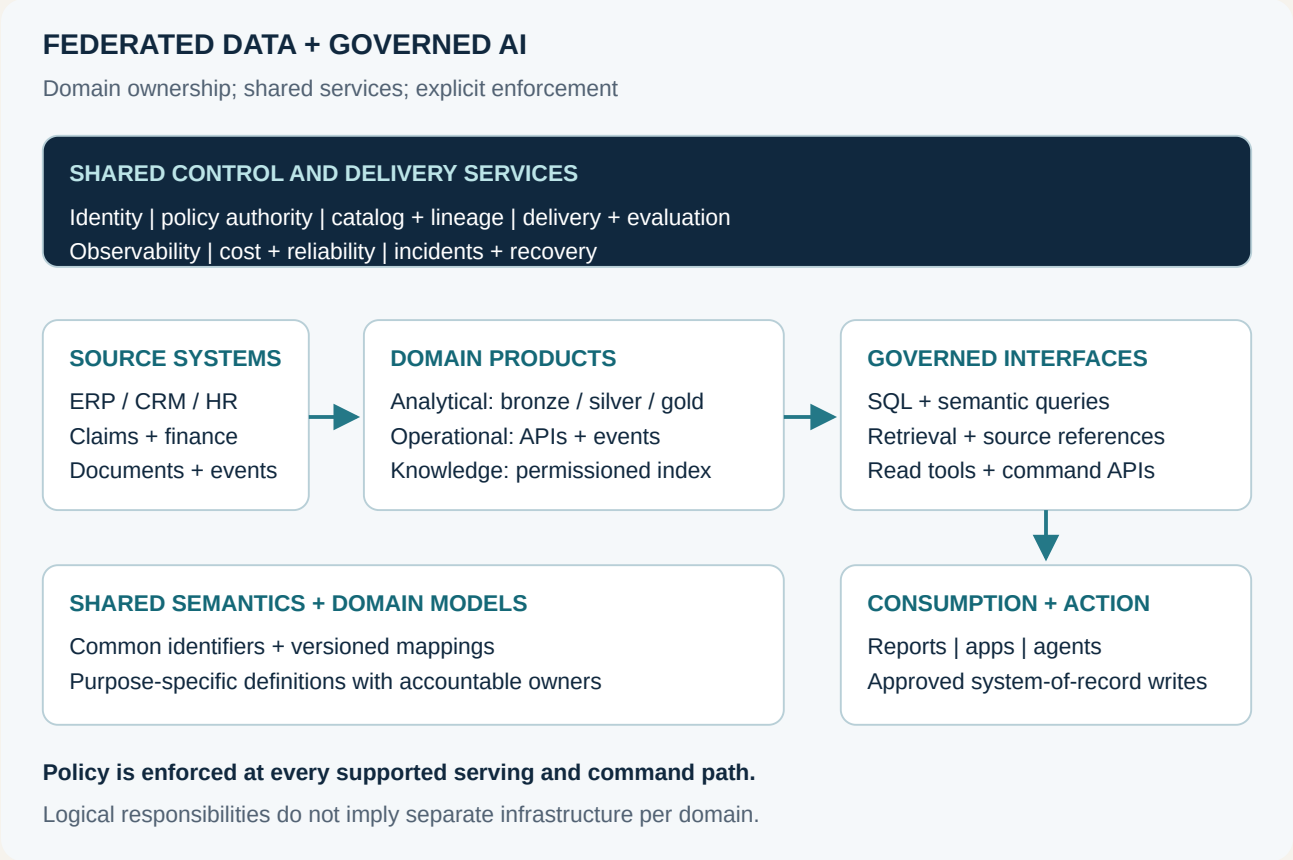
Federation allocates responsibility under common standards. It doesn't require each domain to buy its own stack or maintain separate infrastructure. A platform can be centrally operated while domain teams control their product roadmaps.

PREREQUISITES FOR USEFUL FEDERATION	Start where business ownership, engineering capacity and shared identity already exist. Give support where domains are still building capability. And never hand out operational responsibility without the means to carry it.

02 / REFERENCE ARCHITECTURE

Connect the analytical and operational paths.

Domains publish supported products; shared infrastructure and a common control plane make their operation coherent.



RJL reference design adapted from Richard White's architecture. Shared services govern all product and access paths.

Domains own meaning, pipelines and products. The platform provides shared runtime and control services. Both analytical and operational paths need contracts, accountable owners and tested controls.

ARCHITECTURE IS NOT AUTHORIZATION — Name the policy authority and each serving enforcement point. Test permitted calls, prohibited calls and alternate access paths.

Shared services span analytical, operational and knowledge products. Domain ownership does not require a separate lakehouse. See the architecture update for the complete request and action path.

03 / LAYERS AND INTERFACES

Publish gold. Choose the serving path by how current the answer must be.

A data product is a maintained promise to a defined consumer. Gold is our publishing convention for analytical products; bronze and silver stay inside the pipeline.

LAYER	PURPOSE	WHO MAY USE IT
Bronze / source representation	Preserve source fidelity and provenance for replay and audit.	The pipeline and its engineers, under restricted access and retention.
Silver / validated representation	Apply agreed keys, types, quality rules, deduplication and late-arriving corrections.	Approved engineering and data science work. Not published as a product or listed for business consumers.
Gold / published representation	Serve a defined business purpose through a supported interface.	Consumers, under a contract with an accountable owner and monitored commitments.

Medallion handles changing data well. Silver absorbs corrections and late records [1], and gold republishes them. Operational and knowledge products are products in their own right, published through their own contracts.

What to use when

PRODUCT TYPE AND NEED	USE	THE DATA PRODUCT IS
Analytical: reporting, metrics and models, refreshed in minutes to days	Medallion pipeline	The gold table or view, under contract
Operational: fast, read-only lookups inside an application or agent	Gold plus a low-latency serving copy [6]	The serving interface, fed from gold
Operational, transactional: live state or the ability to write	An API, event stream or change feed from the system of record	The interface contract; writes use a separate command path
Knowledge: documents for AI retrieval	A permission-aware index	The index and its access filter
Any type: data that cannot or should not be copied	Virtualization or federation	The governed view

Keep gold contracts stable while the layers behind them change. For a breaking change, version the interface, tell affected consumers and give them a migration window. The business catalog lists the supported interface. Engineering tables stay visible to engineers, and through lineage.

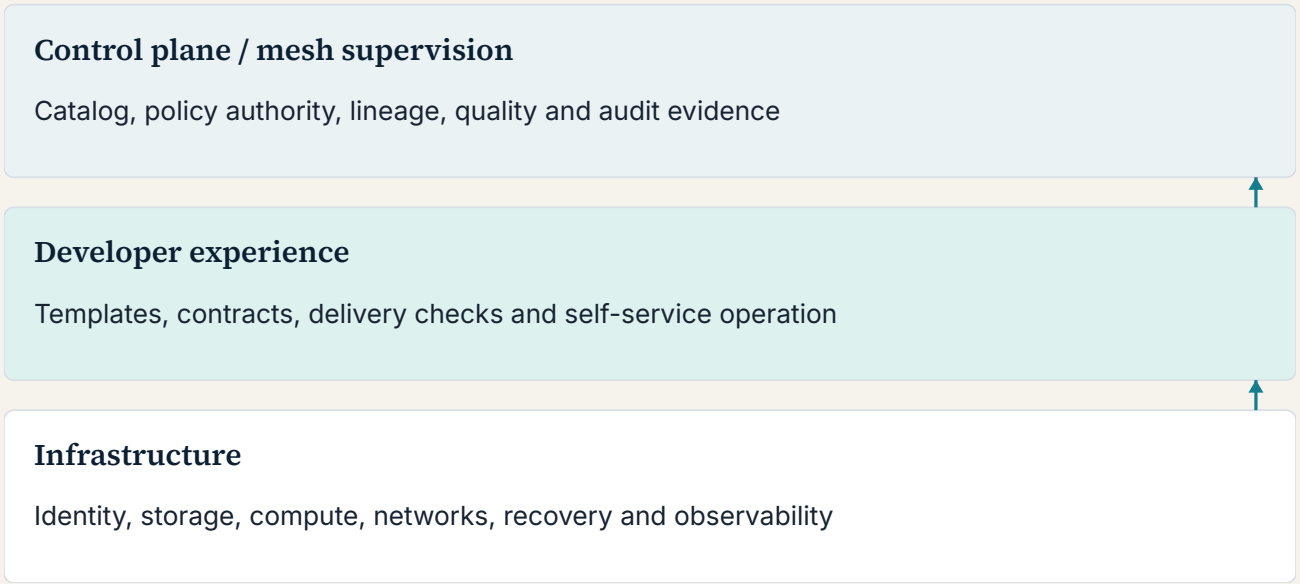
RAW DATA IS NOT UNRESTRICTED

Source fidelity supports audit. It doesn’t authorize broad access or indefinite retention. Apply classification, rights and purpose controls from ingestion onward.

04 / PLATFORM PLANES

Separate runtime, delivery and supervision.

Three planes provide a useful way to organize services and ownership [2].



RJL implementation view. Control plane is used consistently here for mesh supervision services.

Infrastructure services provide reliable resources. Developer-experience services let teams create and operate products through repeatable paths. The control plane collects metadata, coordinates policy and exposes evidence across domains.

Metadata feeds the control plane. It doesn't enforce anything itself. A policy authority evaluates rules, serving runtimes enforce decisions, and tests establish that no supported path bypasses the required control. Keep these responsibilities distinct, even if a vendor combines the components.

ONE AUTHORITY, CONSISTENT ENFORCEMENT	Use common policy definitions and consistent decisions across warehouses, APIs, source queries and retrieval. A federated environment may need several enforcement points. Avoid contradictory rules and uncontrolled alternate paths.

05 / SEMANTICS

Give shared measures an owner and a purpose.

A semantic layer makes definitions explicit. It can't remove every legitimate difference between business contexts.

SEMANTIC RESPONSIBILITY	EXAMPLE
Definitions	Policy in force for servicing versus actuarial valuation, with explicit cutoff and inclusion rules.
Entity and relationship model	Approved party, policy and claim keys; valid cardinality and join behavior.
Serving and change	A versioned metric or query interface; consumer impact and approval for changes.

A governed measure includes grain, units, dates, scope, calculation, source dependencies and an accountable owner. When two functions need different meanings, keep both and publish the reconciliation. Eliminate accidental inconsistency. Make purpose-specific definitions identifiable.

The business owner approves meaning. Domain engineering implements and tests the calculation. The platform supplies semantic tooling, supported access and metadata. A cross-domain measure needs one accountable owner and explicit source agreements.

Evaluate with a representative use

- Can two permitted tools produce the same result for the same metric version and scope?
- Can the business owner approve the definition and reconstruct a material result?
- Can a consumer migrate if the serving tool changes?
- Do identity and policy hold through every supported access route?

KEEP THE BOUNDARY CLEAR — A semantic layer can help consumers ask the right question. It doesn't replace data quality, current context or access enforcement.

06 / SELECTION CRITERIA

Choose with evidence, not a feature list.

Use a short executive screen and a deeper technical evaluation on the same representative workload.

CRITERION	EVIDENCE TO REQUEST
Compatibility	Existing identity, networks, data engines and development workflow work together.
Standards and portability	Contracts and definitions can be exported and consumed elsewhere.
Ownership and change	Business approval, versioning, lineage and consumer impact are recorded.
Policy and identity	Scoped users and workloads receive consistent decisions on all supported paths.
APIs and integration	Interfaces are documented, supportable and testable without manual reconstruction.
Performance and resilience	Latency, concurrency and recovery meet the workload under realistic conditions.
Automation	Provisioning, testing, deployment and rollback can be repeated.
Adoption and support	Consumers can find, access and use services; operating contacts are clear.
Agent readiness	Tools expose narrow contracts, product context and traceable access.

The executive screen is simple. Does the service solve a real consumer need, carry enforceable control, and avoid unacceptable lock-in or run cost? The technical evaluation should include positive tests, denial tests, source failure and recovery.

Use actual representative data shapes and concurrent workloads where permitted. A demonstration with ideal data and a privileged identity proves nothing about production suitability. Record assumptions, known limitations and the cost of compensating controls.

07 / CONTROL PLANE IN ACTION

Contain a source change before it spreads.

Impact analysis helps when it reaches an owner and changes the serving behavior.

Illustrative example: a core system renames a balance field in a nightly extract. The domain pipeline detects the schema difference before publishing a new business-ready version. It doesn't silently reinterpret the incoming value.

STEP	OPERATING BEHAVIOR
Detect	Flag the schema change; retain source/version evidence and stop the affected publication path.
Assess	Use lineage to identify dependent products, measures, applications and agent tools.
Contain	Serve a marked last-good version only within its approved stale-data window; otherwise restrict or stop.
Notify	Alert accountable product owners and affected consumers with impact and expected response.
Fix and record	Update mapping, run tests and reconciliation, republish, and retain the decision record.

The control plane should make owner, version, classification, quality status and critical lineage discoverable. The pipeline and serving services implement the actual failure response, using product-specific policy. A catalog warning alone can't stop an application from acting on stale data.

Measure time to detection, time to identify affected consumers, time to restoration, and confirmation that prohibited calls were blocked. Access automation is useful. The percentage of automated decisions alone, though, tells you nothing about control effectiveness.

EVIDENCE BEYOND THE HAPPY PATH	Exercise source drift, missed input, revoked access, policy-service failure and stale output. Record the behavior each consumer should get.

08 / COPY OR QUERY

Make movement a recorded choice.

Virtualization avoids some copies. Ingestion can provide history, control and resilience. Neither is always the right answer.

QUESTION	WHEN A COPY MAY HELP	WHEN IN-PLACE ACCESS MAY HELP
Need reproducible history?	A controlled snapshot supports audit and restatement.	The source provides authoritative versioned history.
Need reliable performance?	A serving copy isolates concurrency and latency.	Source capacity and access latency meet the workload.
Can policy be enforced?	The destination carries required rights and controls.	The source applies equivalent tested policy.
Must the use survive source failure?	A defined fallback needs a retained copy.	Availability and failure behavior meet the commitment.
Are rights restrictive?	A licensed copy is permitted and governed.	Retention or sharing terms favor controlled in-place use.

For each copy, record purpose, owner, source version, refresh, allowed use, retention and deletion. For in-place access, record source identity propagation, policy behavior, latency limits and outage handling. Don’t assume a virtual query inherits destination policy.

A ledger snapshot may be justified for reproducible period-end reporting. A current consent query may be better served from its authoritative system, provided availability and authorization hold. If you cache consent, make the revocation and expiry behavior explicit.

COUNT ALL OPERATING COST	Include source load, transfer, storage, serving compute, support and synchronization. Zero-copy isn’t zero-cost, and it isn’t automatically safer.

09 / GOLDEN PATHS

Make the safe path reproducible.

A golden path is a supported implementation pattern with tested defaults and explicit exceptions.

TEMPLATE	BUILT-IN RESPONSIBILITIES
Source intake	Provenance, schema checks, source permissions, retention and failure quarantine.
Analytical product	Contract, transformations, quality tests, lineage, publication and monitoring.
Operational API / event feed	Scoped identity, latency and availability measures, idempotency and fallback.
Semantic measure	Business approval, versioned definition, reconciliation and consumer impact.
Agent tool	Narrow input/output contract, context prerequisite, budget, denial tests and trace.

Store templates, policies and configuration in version control. Review changes, test them, and make deployment and rollback repeatable. AI coding assistants can help implement patterns. Generated code still has to pass the same checks and accountable review.

Keep the paved path adaptable. A domain with a justified need can use an approved exception, with compensating controls, an owner and an expiry. Feed repeated exceptions into platform improvement; don’t let permanent workarounds pile up.

Track template adoption, time to the first working increment, release failure and support burden. A complex template nobody can use isn’t a successful standard. Provide examples, coaching and documented troubleshooting.

THE PLATFORM PROMISE — A team can create a supported product without rebuilding identity, deployment, monitoring and baseline policy from scratch.

10 / CAPABILITY MAP

Connect tools to service responsibilities.

An architecture inventory is more useful when it shows what each capability does for a consumer.

STAGE	CAPABILITIES	RESPONSIBILITY
Source	Operational systems, documents and licensed feeds.	Source authority, rights, event meaning and change notice.
Integrate	Batch, change capture, events and controlled queries.	Delivery checks, provenance, synchronization and replay.
Store and process	Analytical history, live state, transformations and indexing.	Data condition, retention, reliable computation and recovery.
Access and serve	SQL, APIs, events, semantic views and permitted retrieval.	Supported interface, identity, policy and commitments.
Consume	Reports, workflows, models and agents.	Purpose, interpretation, outcome and consumer controls.

Operation, security and governance run across these stages. Name service and product owners; a list of vendors isn't enough. A platform team can operate shared infrastructure while a domain keeps responsibility for its product definition and runbook.

Use the map to find duplicate integrations, hidden extracts and control gaps. Consolidate tools only when the replacement meets business needs and consumers have a safe migration path.

Common enterprise services

- Catalog and critical lineage; reference identifiers and definitions.
- Policy authority, identity, audit records and exception management.
- Delivery automation, observability, cost visibility and recovery.

START WITH CAPABILITIES ALREADY PRESENT	Reuse enterprise services that work. A new architecture shouldn't assume a wholesale replacement of the current stack.

Give AI a governed runtime.

Model access is one part of the platform. Tool behavior, evidence and operating limits matter just as much.

A model gateway can manage approved providers, versions, credentials, routing, usage and cost. An agent runtime coordinates task scope, context, tools, approval and trace. Neither component alone proves an answer correct or a workflow safe.

AI RMF FUNCTION [3]	RJL IMPLEMENTATION EVIDENCE
Govern	Accountable owner, approved uses, provider controls, escalation and risk-based release authority.
Map	Documented task, users, data rights, affected people and foreseeable failures.
Measure	Representative evaluations, negative policy tests, quality checks and monitored production behavior.
Manage	Human intervention, budgets, restriction/stop controls, incident response and recovery.

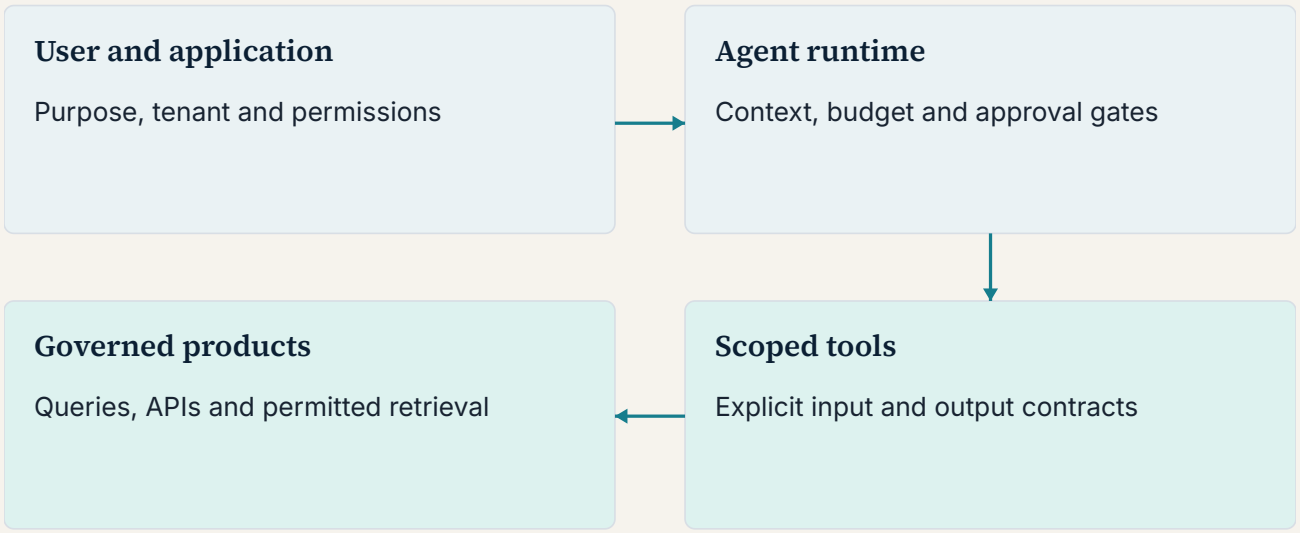
NIST AI RMF is voluntary guidance, and its Generative AI Profile identifies additional risks [4]. The table adapts it for implementation; it makes no conformity claim. For the actual use, evaluate factual error, inappropriate disclosure, prompt injection, unsupported actions and third-party dependency.

Maintain evaluation cases with expected evidence and permitted behavior. Re-evaluate whenever models, prompts, tools, product definitions or rights change. A result with a source citation can still misread that source, so answer quality needs evaluation beyond whether a citation is present.

OPERATE WITHIN A DEFINED ENVELOPE	Set cost, rate and action limits. Require approval where the task can materially affect customers, money or regulated decisions. Provide a tested fallback and a responsible operator.

Expose narrow tools over governed products.

A tool protocol provides connectivity. Authorization and context need explicit implementation.



Trace, evaluate and enforce identity and policy across every call.

Illustrative access path. MCP is one possible protocol for exposing tools and resources [5].

Carry the user scope and workload identity through the call chain. Never treat an agent as entitled to a shared privileged credential. The runtime fetches current product context and validates the approved task before invoking the tool.

Tools should define inputs, outputs, permissions, timeouts, failure responses and allowed actions. Enforce access at the serving service. Document whether a tool queries data, retrieves content or changes state, and apply stronger approval and recovery where it can change outcomes.

Use a knowledge graph where relationships justify it. It isn't mandatory. Apply the same rights, freshness and lineage expectations to any representation.

PROVE THAT BYPASS FAILS	Test missing context, mismatched product versions, revoked rights, out-of-scope tenant data and prohibited detail queries. Protocol connectivity doesn't make these tests optional.

Release only when the minimum evidence exists.

Build incrementally, while meeting the readiness bar from the first production product.

SEQUENCE	EVIDENCE REQUIRED
1. Bound the outcome	Owner, consumer, critical data, supported interface and available delivery capacity.
2. Establish minimum controls	Classification, identity, tested access, critical quality, essential lineage and run ownership.
3. Publish the first product	Approved contract, release evidence, monitoring, fallback and representative use.
4. Extend shared services	More catalog/lineage coverage, templates, policy automation and semantic reuse.
5. Add operational and agent uses	Use-specific latency, rights, evaluations, approval gates and traceability.
6. Consolidate and improve	Observed reuse, sustainable run cost and safe retirement of duplicate paths.

Essential lineage and quality evidence come before certification. Later work expands coverage and automation; it doesn't bolt these prerequisites on after the first release. An operational path or an agent can appear earlier, when the first bounded use needs it and its own readiness criteria are met.

In a smaller firm, the platform may be one managed cloud stack and two engineers. The sequence still holds. Each step is smaller, and the shared services are configuration and conventions instead of separate teams.

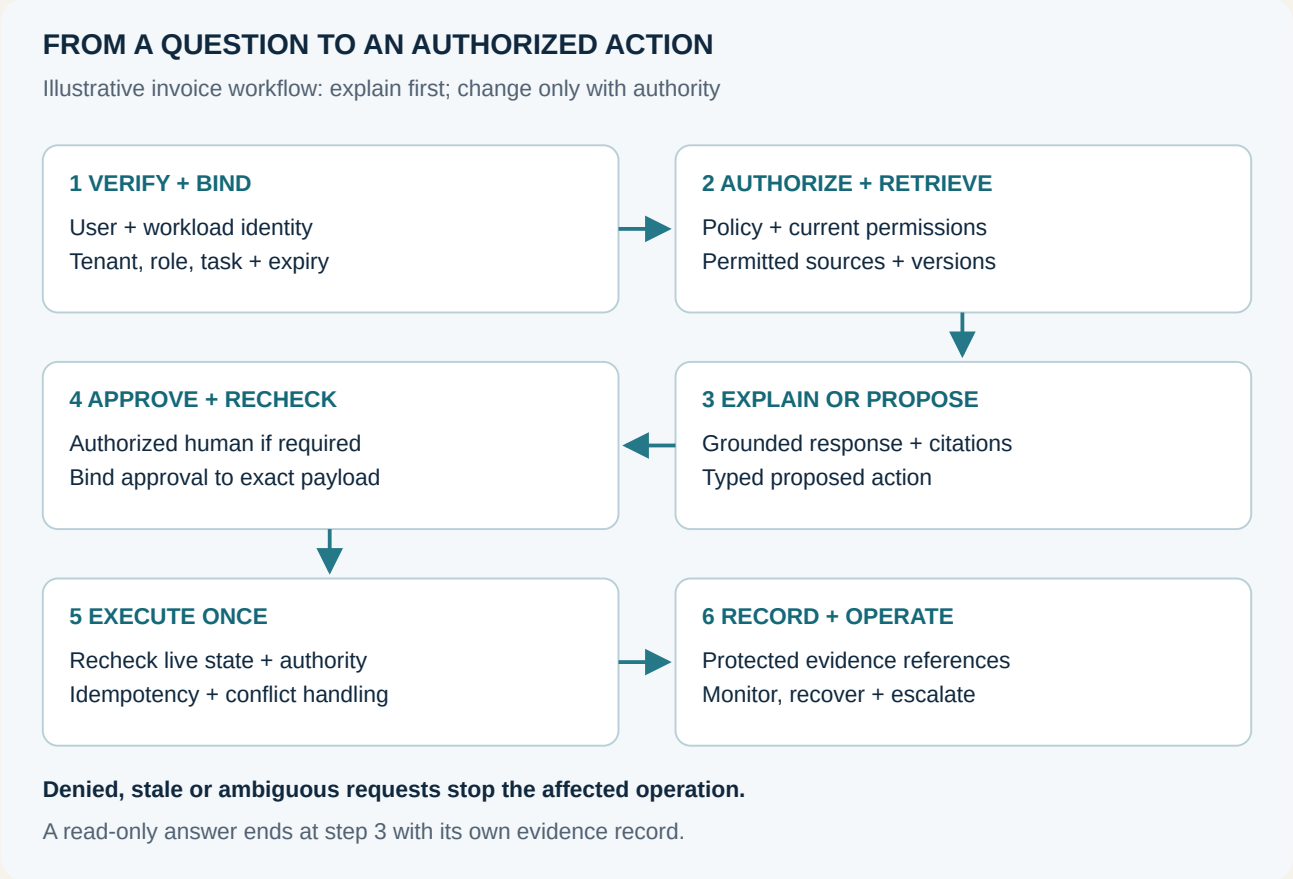
Review time to usable access, delivery lead time, commitment breaches, policy failures, consumer adoption and build/run cost. Tie platform investment to measured bottlenecks and business outcomes.

A PRACTICAL STARTING POINT	Choose one outcome, one or two accountable domains and the smallest supported serving path. Use existing services where they work. Expand when the evidence justifies the next investment.

ARCHITECTURE UPDATE / REQUEST VIEW

Make the authority chain visible.

The model explains or proposes. Trusted services decide what may be read or changed.



Illustrative control sequence. Retrieved content and model output never confer authority to execute a command.

An employee asks why an invoice is held. The application verifies identity, binds the tenant and legal entity, and retrieves permitted invoice and policy records. The answer cites its sources. A request to change the invoice follows a separate command route, with approval where required, a fresh state check and a recorded result.

ARCHITECTURE UPDATE / OPERATING CONTRACT

Prove the connections, not just the boxes.

Use the invoice example to test the whole design before extending it to a regulated decision.

BOUNDARY	IMPLEMENTATION AND RELEASE EVIDENCE
Domain ownership	Name the business meaning owner, engineering owner, contract version and supported interface. Shared infrastructure is compatible with domain ownership.
Context and identity	Verify user and workload identities. Resolve department and legal entity from authoritative systems when needed; confirm ambiguous intent with the user.
Read access	Enforce tenant, purpose and document/row permissions at retrieval and serving. Permission changes must reach indexes and caches within an agreed window.
Write access	Validate typed inputs; bind approval to the exact payload and approver authority. Recheck permissions and record version at execution. Never retry an uncertain write blindly.
Operation	Retain protected evidence references, set latency and cost budgets, and exercise denial, stale data, outage and recovery paths.

For claims, the same pattern can assemble permitted evidence and route a recommendation to an authorized reviewer. Adjudication or payment is a separate capability with its own policy and release criteria. A lean firm may implement these responsibilities in one managed stack.

Design references checked 29 October 2026: Data mesh principles · Document-level access control · RAG security. These additions are RJL implementation recommendations, not a vendor certification or a compliance determination.

ARCHITECTURE UPDATE / KNOWLEDGE AND ROUTING

Bring the right evidence to the task.

The intelligence layer resolves meaning and routes permitted requests. It does not need to become a copy of every source system.

QUESTION OR ASSET	SERVING DECISION AND BOUNDARY
What does invoice balance mean?	Resolve the versioned definition, currency, valuation time and relevant legal entity. Use shared identifiers and explicit mappings between domain models.
Why is this invoice held?	Read the current hold through a governed source API and retrieve the permitted policy passages. Combine live facts and document evidence without treating either as an instruction to the agent.
What changed across the portfolio?	Use a contracted analytical product or semantic query with the right grain, period and freshness. State any delay relative to the system of record.
Can we change the hold?	Route to a narrowly scoped command tool. Retrieval, a graph relationship or a plausible explanation cannot authorize the write.
Should the content be copied?	Choose ingestion, controlled indexing or source queries based on rights, latency, resilience, freshness and cost. Virtualization does not remove those obligations.

A source-aligned product publishes a stable contract around one domain capability. A derived product combines or transforms supported products for a consumer purpose and owns the new meaning, lineage and service commitments. Neither label determines which physical medallion layer must be exposed.

See Data Products in Practice for the staged invoice construction example and its foundational-to-derived product map.

Knowledge preparation can require OCR, parsing, chunking, permission metadata and hybrid indexing. A knowledge graph is useful where relationship reasoning warrants its maintenance cost; it is optional. Agent memory is a separate derived store with tenant scope, provenance, retention and deletion rules. Recheck its contents before reuse; yesterday's permitted retrieval does not authorize today's disclosure.

Design references checked 29 October 2026: Data mesh principles · Document-level access control · RAG security. These additions are RJL implementation recommendations, not a vendor certification or a compliance determination.

Sources and scope.

Public frameworks inform the design. The implementation choices, checklists and examples are RJL recommendations, drawn from Richard White's practice.

[1] *Medallion lakehouse architecture*. Databricks documentation. Bronze, silver and gold are analytical design conventions; the broader architecture here is vendor-neutral. docs.databricks.com / medallion architecture

[2] *Data Mesh Principles and Logical Architecture*. Zhamak Dehghani, December 2020. Domain ownership and the platform planes inform this implementation design. martinfowler.com / data mesh principles

[3] *AI Risk Management Framework 1.0*. NIST AI 100-1, January 2023. Voluntary framework with Govern, Map, Measure and Manage functions. nist.gov / AI RMF 1.0

[4] *Generative AI Profile*. NIST AI 600-1, July 2024. Referenced for use-specific generative-AI risk evaluation. nist.gov / Generative AI Profile

[5] *Model Context Protocol*. Official specification. MCP supports tool and resource connectivity; the surrounding controls are architectural responsibilities. modelcontextprotocol.io / specification

[6] *Serve lakehouse data with synced tables*. Databricks documentation, accessed October 2026. One example of a low-latency, read-only serving copy of gold data for applications; Snowflake hybrid tables serve low-latency application reads and writes, though not as a synced copy. docs.databricks.com / Lakebase synced tables

Sources reviewed 10 October 2026; each is linked. Examples are illustrative; proposed measures and service commitments need an agreed baseline and risk assessment. This paper does not establish regulatory compliance or reproduce a licensed assessment standard. Companion papers at rjlsol.com/whitepapers.

Read the series

PAPER	WHAT IT COVERS
The Data and AI Operating Model	Ownership, teams, decision rights and funding.
Data Products in Practice	Boundaries, contracts, quality and certification.
The Federated Data Platform (this paper)	Shared services, access paths, resilience and agents.
Context Before Query	Context, authorization and enforceable policy for AI.
AI Governance Evidence	For regulated firms: what bank and insurance reviewers need to inspect.

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.

Bring one use case and the current source-to-consumer architecture. RJL Solutions helps identify reusable products, the shared services worth building first and the controls each serving path must prove.

OPERATING MODEL **The Data and AI Operating Model**
Ownership, teams, decision rights and funding.

PRODUCTS **Data Products in Practice**
Boundaries, contracts, quality and certification.

GOVERNANCE **Context Before Query**
Context, authorization and enforceable policy for AI.

RJL SOLUTIONS

WEB rjlsol.com

EMAIL rwhite@rjlsol.com

Examples are illustrative and describe no client engagement or client result. Measures, timings and service commitments are starting points that need an agreed baseline. Apply the design to your business purpose, existing controls and any obligations that apply to you.