ManChine

AI TECHNOLOGY

WHITE PAPER 005

The RiCo Runtime Platform

Operationalizing Runtime Integrity Control

RiCo

Runtime Integrity Control

Richard Colimon

Founder & Chief Executive Officer

ManChine AI Technology LLC

ManChine AI Technologies

Governance-layer software for AI runtime integrity, continuity, and consequential governance.

Humans First.
Continuity Always.
Architecture Must Survive.

RiCo

Runtime Integrity Control

Abstract

The first four papers established the conceptual foundations of Runtime Integrity Control (RiCo).

White Paper 001 introduced Runtime Integrity Control as a governance architecture for preserving the legitimacy of consequential execution.

White Paper 002 identified the Runtime Execution Boundary (REB) as the location where legitimacy must be continuously evaluated before consequences become binding.

White Paper 003 defined the Runtime Governance Stack—the architectural components responsible for maintaining runtime legitimacy.

White Paper 004 described the Runtime Protocol that enables those components to exchange governance evidence, authority, policy, and execution context while preserving reconstructability.

This paper turns from architecture to implementation.

It introduces the **RiCo Runtime Platform**—the operational platform that realizes these architectural concepts as deployable software.

The platform consists of interoperable runtime services including the RiCo Runtime Engine, the Runtime Execution Boundary (REB) SDK, the Runtime Governance API, the Governance Protocol SDK, the Evidence Graph, and the Runtime Trust Dashboard.

Together these components enable governance to operate continuously during execution rather than exclusively through retrospective audit or static policy enforcement.

This paper presents the functional responsibilities of each platform component, explains how they cooperate to preserve runtime legitimacy, and outlines a practical path toward interoperable governance infrastructures capable of operating across heterogeneous AI systems.

The objective is not to replace existing AI platforms.

It is to provide the governance runtime capable of preserving legitimate consequence regardless of the underlying model, application, or execution environment.

1. Introduction

The development of Runtime Integrity Control (RiCo) has proceeded through a deliberate architectural progression.

Each paper in this series has addressed a distinct question required to establish governance as a first-class runtime capability rather than a post-execution auditing function.

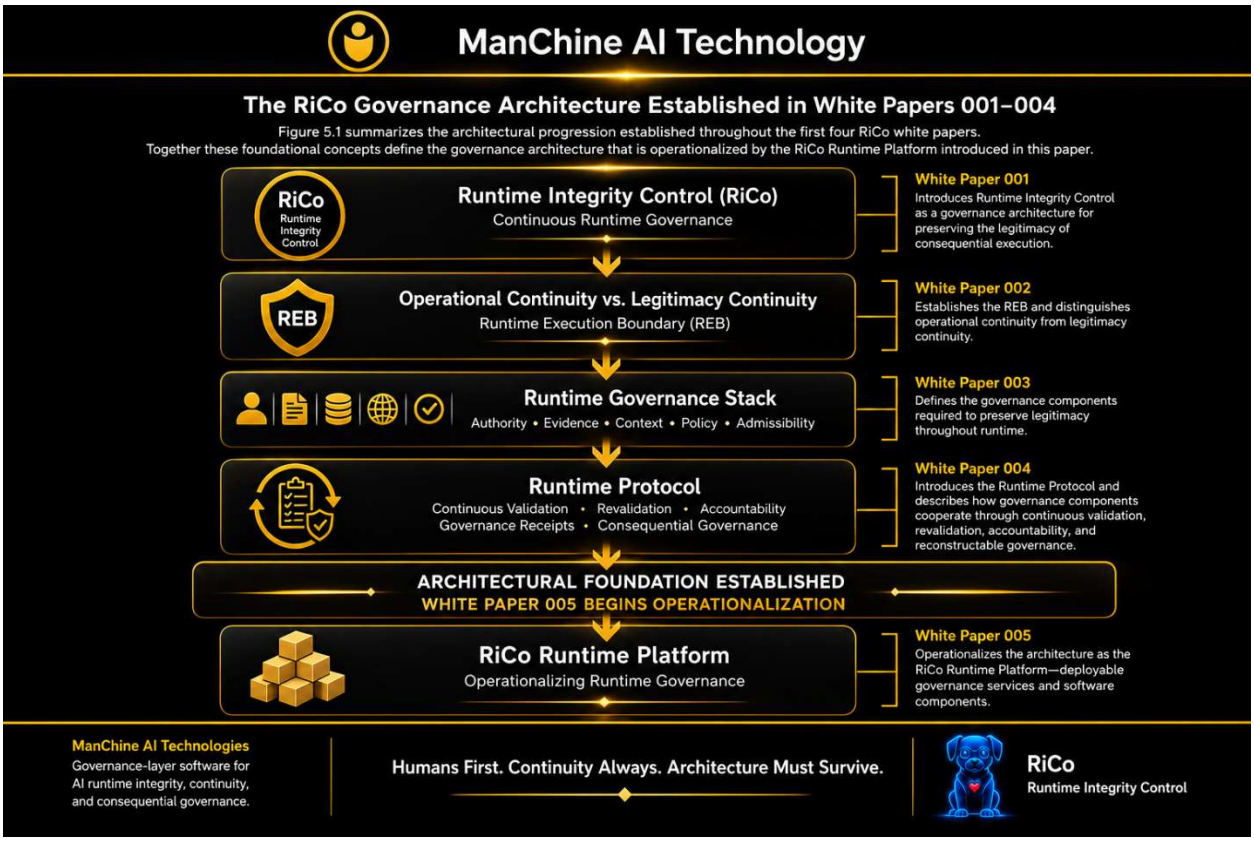
White Paper 001 introduced **Runtime Integrity Control (RiCo)** as a governance architecture designed to preserve the legitimacy of consequential execution.

White Paper 002 established the **Runtime Execution Boundary (REB)** as the architectural location where governance decisions must be evaluated before execution produces consequential outcomes. It distinguished operational continuity from legitimacy continuity, demonstrating that systems may continue operating while losing the conditions that justify the consequences they produce.

White Paper 003 defined the **Runtime Governance Stack**, identifying the architectural components required to evaluate evidence, authority, policy, context, and admissibility during execution.

White Paper 004 introduced the **Runtime Protocol**, describing how those components exchange governance information while preserving reconstructability, traceability, and runtime legitimacy.

Together, these four papers establish a complete conceptual architecture for runtime governance.



Summarizes this progression and illustrates how the foundational concepts developed across White Papers 001–004 operate together as a unified governance architecture.

“The RiCo Governance Architecture Established in White Papers 001–004”

The natural next question is no longer **what** runtime governance is, nor **where** it occurs, nor **which** components participate, nor **how** they communicate.

The remaining question is practical:

How is this architecture deployed?

This paper answers that question by introducing the **RiCo Runtime Platform**.

Rather than presenting a new governance model, the Runtime Platform operationalizes the architecture established in the previous papers as deployable software services. These services provide the runtime capabilities necessary to evaluate legitimacy continuously, enforce governance policies consistently, preserve evidence throughout execution, and expose standardized interfaces that allow heterogeneous AI systems to participate within a common governance framework.

The platform consists of interoperable runtime components, including the **RiCo Runtime Engine**, the **Runtime Execution Boundary (REB) SDK**, the **Runtime Governance API**, the **Governance Protocol SDK**, the **Evidence Graph**, and the **Runtime Trust Dashboard**.

Collectively, these components transform runtime governance from an architectural concept into an operational capability that can be integrated into enterprise software, autonomous agents, distributed systems, and future governance-aware AI infrastructures.

This paper therefore marks a transition from architectural definition to platform realization. It describes not only the components of the RiCo Runtime Platform, but also how they cooperate to preserve legitimate consequence throughout execution while remaining interoperable across diverse technologies and deployment environments.

2. Why a Runtime Platform?

The preceding papers established the architectural principles required for continuous runtime governance. They defined the Runtime Execution Boundary (REB), the Runtime Governance Stack, and the Runtime Protocol through which governance information is exchanged.

However, architecture alone does not execute.

An architecture describes **what** must exist.

A platform provides **how** those capabilities operate continuously within running systems.

This distinction is fundamental.

Artificial intelligence has rapidly developed sophisticated execution platforms capable of generating recommendations, making predictions, invoking tools, and orchestrating increasingly autonomous behavior.

Yet comparatively little infrastructure exists for continuously evaluating whether those actions remain legitimate as authority, evidence, policy, and operational context evolve.

Execution platforms optimize intelligence.

Governance platforms preserve legitimacy.

These are complementary responsibilities.

One determines **what can be done.**

The other determines **what should remain admissible.**

Runtime Integrity Control (RiCo) therefore requires more than a conceptual framework.

It requires an operational platform capable of evaluating legitimacy continuously throughout execution rather than exclusively before deployment or after consequence has already occurred.

The RiCo Runtime Platform fulfills this role.

Rather than replacing existing AI frameworks, enterprise applications, or orchestration systems, it operates alongside them as an independent governance layer.

Its responsibility is not to generate decisions.

Its responsibility is to determine whether proposed execution remains admissible before consequences become binding.

This separation preserves an important architectural principle established throughout this series:

Execution and governance are distinct runtime responsibilities.

Execution seeks successful completion.

Governance seeks legitimate consequence.

The RiCo Runtime Platform enables these responsibilities to cooperate without requiring either to absorb the responsibilities of the other.

As intelligent systems continue to become more autonomous, distributed, and interconnected, governance must likewise become operational, interoperable, and continuously available.

For this reason, the RiCo Runtime Platform is presented not as another application, but as foundational runtime infrastructure capable of supporting heterogeneous AI systems, enterprise software, autonomous agents, and future governance-aware execution environments.

3. RiCo Runtime Platform Architecture

The RiCo Runtime Platform is the operational realization of the governance architecture established throughout the preceding papers.

Rather than existing as a single executable application, the platform consists of a collection of interoperable runtime services that collectively preserve the legitimacy of consequential execution.

Each component performs a distinct governance responsibility while cooperating through the Runtime Protocol and operating at the Runtime Execution Boundary (REB).

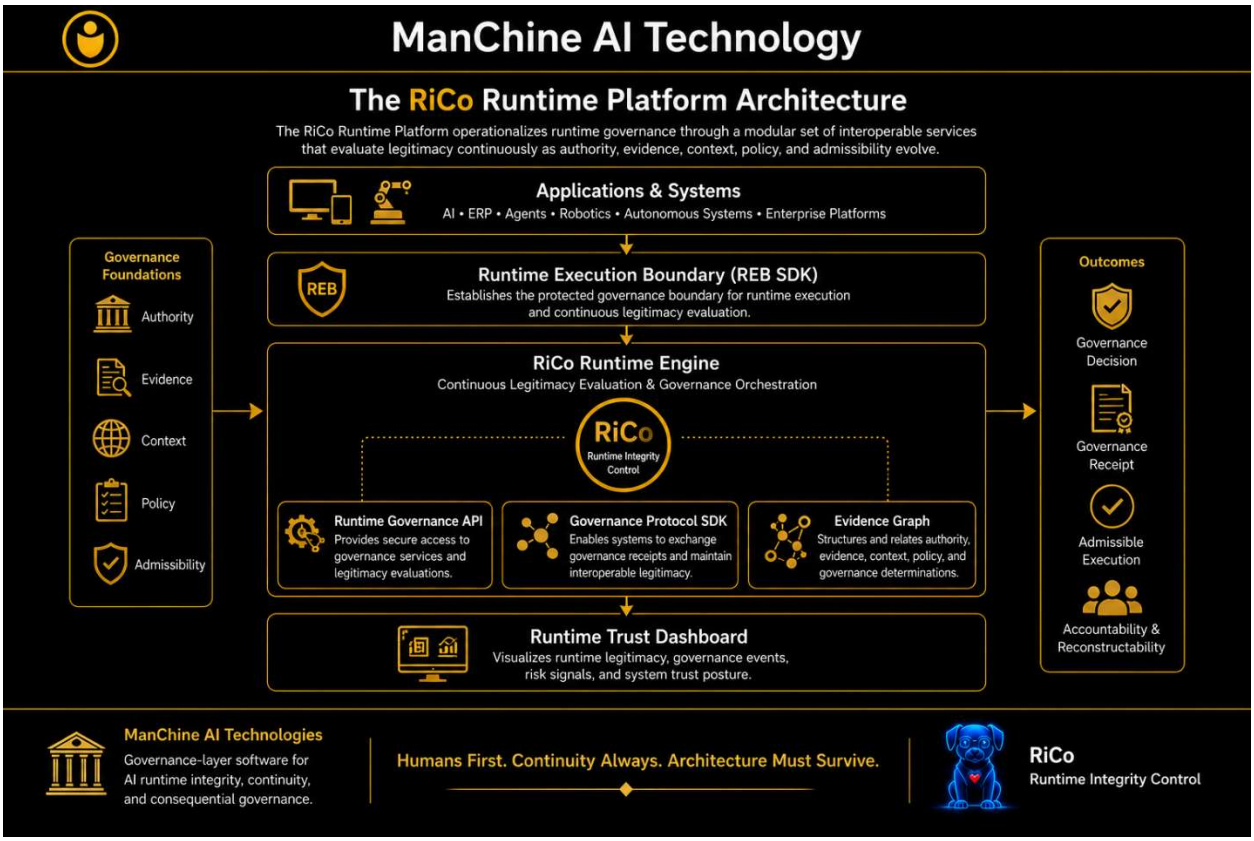
This modular architecture allows organizations to adopt individual capabilities incrementally while maintaining interoperability across heterogeneous AI systems, enterprise applications, autonomous agents, and distributed execution environments.

Unlike traditional governance solutions that rely primarily upon static policies, identity management, or retrospective auditing, the RiCo Runtime Platform operates continuously during execution.

Every proposed consequential action may therefore be evaluated against current evidence, authority, policy, operational context, and admissibility before consequences become binding.

The platform is organized into six principal runtime services.

3.1 Platform Components



Each component addresses a different aspect of runtime governance while remaining independent of any specific artificial intelligence model, enterprise application, or execution framework.

Together they provide the operational capabilities required to preserve runtime legitimacy throughout consequential execution.

RiCo Runtime Engine

The Runtime Engine serves as the core governance processor of the platform.

It evaluates proposed execution against governance evidence, authority, policy, operational context, and admissibility before execution is permitted to produce binding consequences.

The engine does not generate decisions.

It evaluates whether proposed decisions remain legitimate.

Runtime Execution Boundary (REB) SDK

The REB SDK enables existing software systems to integrate directly with the Runtime Execution Boundary.

Applications invoke the SDK whenever execution reaches a consequential transition requiring governance evaluation.

This allows existing systems to adopt RiCo without replacing their underlying execution engines.

Runtime Governance API

The Runtime Governance API exposes governance capabilities through standardized service interfaces.

Applications, enterprise platforms, autonomous agents, and distributed services may request governance assessments without requiring direct integration with the Runtime Engine itself.

This enables Governance as a Service across heterogeneous computing environments.

Governance Protocol SDK

The Governance Protocol SDK implements the Runtime Protocol defined in White Paper 004.

It standardizes the exchange of governance evidence, authority assertions, policy references, admissibility assessments, governance receipts, and execution outcomes across participating systems.

Through standardized protocol implementation, independently developed systems remain interoperable while preserving reconstructability.

Evidence Graph

The Evidence Graph maintains the relationships between observations, evidence, authority, policies, governance assessments, and consequential outcomes.

Rather than storing isolated records, it preserves governance lineage across the complete execution lifecycle.

This allows every consequential decision to be reconstructed from its originating evidence through its resulting consequence.

Runtime Trust Dashboard

The Runtime Trust Dashboard provides real-time visibility into governance activity across the platform.

Administrators, auditors, and governance officers can observe legitimacy evaluations, evidence completeness, policy enforcement, authority chains, and execution outcomes as they occur.

The dashboard emphasizes operational trust through transparency rather than retrospective reporting.

3.2 Architectural Cooperation

Although each component performs a distinct responsibility, none operates independently.

The Runtime Engine evaluates legitimacy.

The REB SDK establishes the execution boundary.

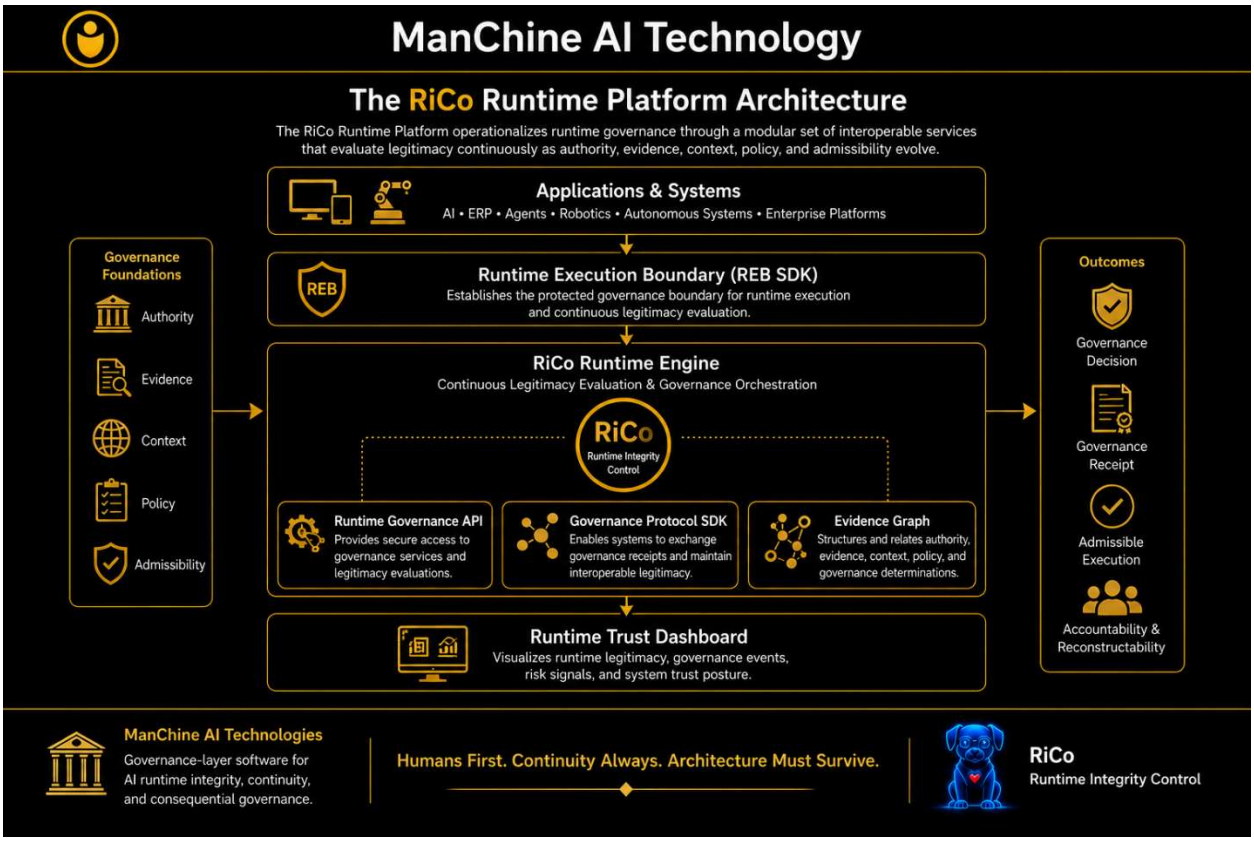
The Governance API exposes governance services.

The Governance Protocol SDK enables interoperable communication.

The Evidence Graph preserves reconstructable governance lineage.

The Runtime Trust Dashboard provides operational visibility into the complete governance lifecycle.

Together these components transform Runtime Integrity Control from an architectural framework into deployable runtime infrastructure capable of preserving legitimate consequence across heterogeneous intelligent systems.


ManChine AI Technology
The RiCo Runtime Platform Architecture
The RiCo Runtime Platform operationalizes runtime governance through a modular set of interoperable services that evaluate legitimacy continuously as authority, evidence, context, policy, and admissibility evolve.
Applications & Systems
AI • ERP • Agents • Robotics • Autonomous Systems • Enterprise Platforms
Governance Foundations
Authority
Evidence
Context
Policy
Admissibility
REB
Runtime Execution Boundary (REB SDK)
Establishes the protected governance boundary for runtime execution and continuous legitimacy evaluation.
RiCo Runtime Engine
Continuous Legitimacy Evaluation & Governance Orchestration
RiCo
Runtime Integrity Control
Runtime Governance API
Provides secure access to governance services and legitimacy evaluations.
Governance Protocol SDK
Enables systems to exchange governance receipts and maintain interoperable legitimacy.
Evidence Graph
Structures and relates authority, evidence, context, policy, and governance determinations.
Runtime Trust Dashboard
Visualizes runtime legitimacy, governance events, risk signals, and system trust posture.
Outcomes
Governance Decision
Governance Receipt
Admissible Execution
Accountability & Reconstructability
ManChine AI Technologies
Governance-layer software for AI runtime integrity, continuity, and consequential governance.
Humans First. Continuity Always. Architecture Must Survive.
RiCo
Runtime Integrity Control

4. RiCo Runtime Engine

The **RiCo Runtime Engine** is the core operational component of the RiCo Runtime Platform.

Its responsibility is to continuously evaluate whether proposed execution remains legitimate before consequential effects become binding.

Unlike traditional policy engines that simply evaluate static rules, the Runtime Engine performs continuous legitimacy evaluation by considering multiple governance dimensions simultaneously, including authority, evidence, policy, operational context, and admissibility.

The Runtime Engine does not execute business logic.

It does not generate recommendations.

It does not make autonomous decisions.

Instead, it governs whether proposed execution satisfies the conditions required to produce legitimate consequence.

This distinction preserves the architectural separation established throughout the previous papers between execution and governance.

Execution determines **what will occur.**

The Runtime Engine determines **whether it should occur.**

4.1 Runtime Legitimacy Evaluation

Every consequential execution request arriving at the Runtime Execution Boundary (REB) is evaluated by the Runtime Engine.

Rather than relying upon a single approval rule or static authorization check, the Runtime Engine performs a multidimensional governance assessment.

Typical evaluation dimensions include:

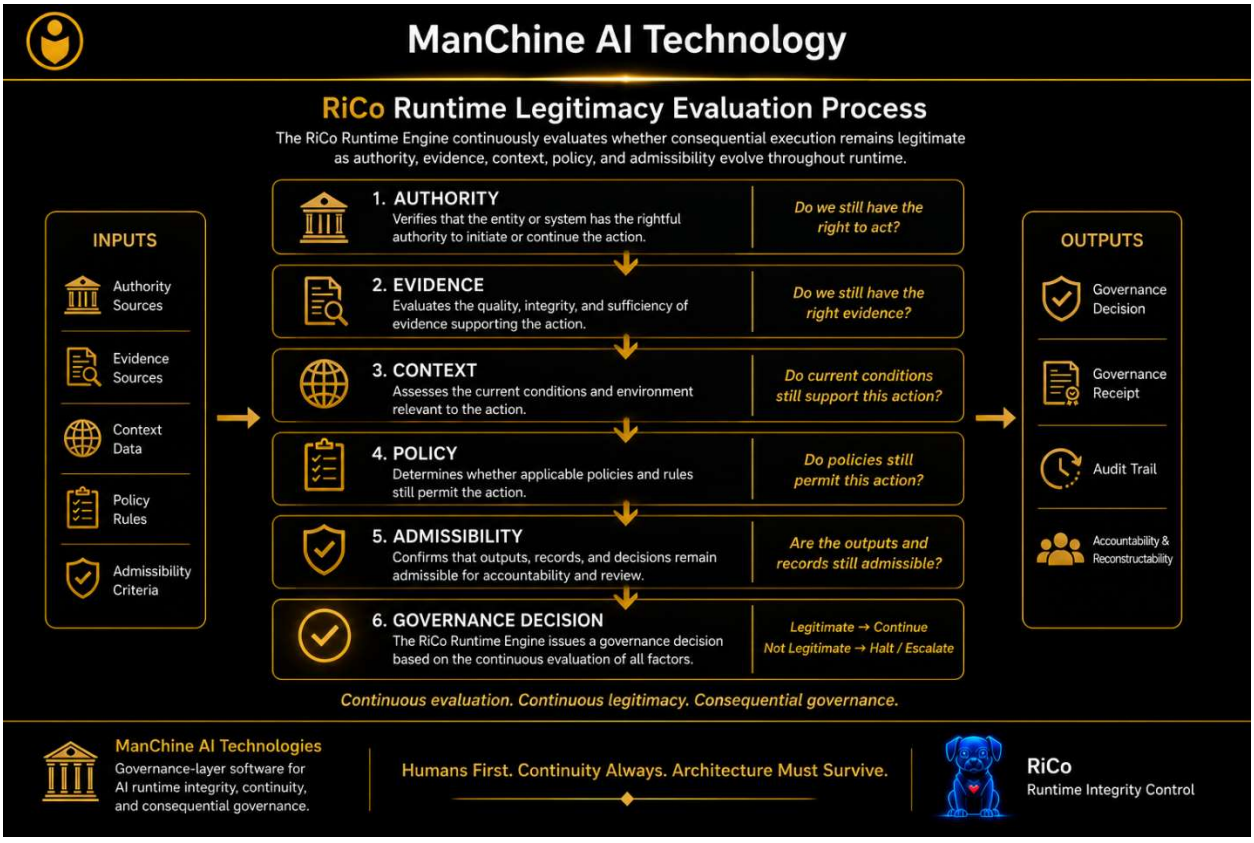
- Authority
- Applicable governance policy
- Supporting evidence
- Operational context
- Requested consequence

- Admissibility status
- Previous governance history
- Runtime observations
- Applicable regulatory requirements
- Organizational governance constraints

The evaluation produces a governance determination before execution is permitted to continue.

4.2 Governance Evaluation Lifecycle

Every governance request follows a consistent evaluation lifecycle.



This lifecycle ensures that every consequential action is evaluated through the same governance process regardless of whether execution originated from a human operator, an autonomous AI agent, an enterprise application, or another distributed system.

4.3 Continuous Governance

The Runtime Engine does not perform governance only once.

Governance remains active throughout execution.

As authority changes, evidence evolves, policies are updated, operational context shifts, or new observations become available, the Runtime Engine may re-evaluate whether continued execution remains legitimate.

This capability reflects one of the central principles established in White Paper 002:

Operational continuity does not necessarily imply legitimacy continuity.

A process may continue executing while the conditions that originally justified its consequences have materially changed.

The Runtime Engine continuously evaluates these changing conditions to preserve legitimacy throughout the execution lifecycle.

4.4 Governance Outcomes

Each evaluation produces a standardized governance outcome.

These outcomes represent the current admissibility status of the proposed execution rather than the success or failure of the underlying business process.

Typical governance outcomes include:

Outcome	Description
Allow	Execution satisfies all governance requirements and may proceed.
Review Required	Additional human governance assessment is required before execution may continue.
Hold	Execution is temporarily suspended pending additional evidence, authority, or policy clarificat

Deny	Execution is not admissible and may not produce binding consequence.

Separating these governance outcomes from operational execution preserves a clear distinction between system functionality and governance legitimacy.

4.5 Architectural Independence

The Runtime Engine remains independent of any particular artificial intelligence model, enterprise application, orchestration framework, or programming language.

Its purpose is not to understand how execution is implemented.

Its purpose is to evaluate whether execution remains legitimate.

Consequently, the Runtime Engine may govern:

- Autonomous AI agents
- Large Language Models (LLMs)
- Enterprise Resource Planning (ERP) systems
- Healthcare platforms
- Financial systems
- Robotics
- Industrial automation
- Distributed cloud services
- Multi-agent systems
- Future intelligent execution environments

This architectural independence enables Runtime Integrity Control to function as a universal governance capability rather than an application-specific solution.

Closing Thought

I think this section should end with a sentence that captures the philosophy of the Runtime Engine:

The RiCo Runtime Engine does not decide what intelligent systems can do. It continuously evaluates whether the consequences of what they propose remain legitimate.

5. Runtime Execution Boundary (REB) SDK

The **Runtime Execution Boundary (REB) SDK** provides the standardized integration layer through which external systems participate in Runtime Integrity Control (RiCo).

While the RiCo Runtime Engine performs continuous legitimacy evaluation, the REB SDK establishes the architectural connection between executing systems and the governance services responsible for determining whether consequential execution remains admissible.

Rather than replacing existing applications, the REB SDK enables them to expose consequential execution requests to the RiCo Runtime Platform before consequences become binding.

This architectural approach preserves a fundamental principle established throughout the RiCo white paper series:

Execution remains the responsibility of the executing system.

Governance remains the responsibility of Runtime Integrity Control.

The REB SDK provides the interface between these responsibilities.

5.1 Purpose of the REB SDK

Every intelligent system eventually reaches moments where execution produces meaningful consequence.

Examples include:

- approving a financial transaction
- issuing an autonomous command
- modifying a medical treatment plan
- authorizing a robotic action
- deploying software into production
- releasing sensitive information
- granting operational authority
- committing a consequential state transition

The REB SDK enables these execution points to become Runtime Execution Boundaries.

Rather than proceeding directly to consequence, execution pauses momentarily while the Runtime Engine evaluates legitimacy.

This evaluation occurs before consequence becomes authoritative.

5.2 Standardized Integration

The REB SDK is intentionally platform-independent.

It is designed to integrate with heterogeneous execution environments without requiring organizations to redesign existing software architectures.

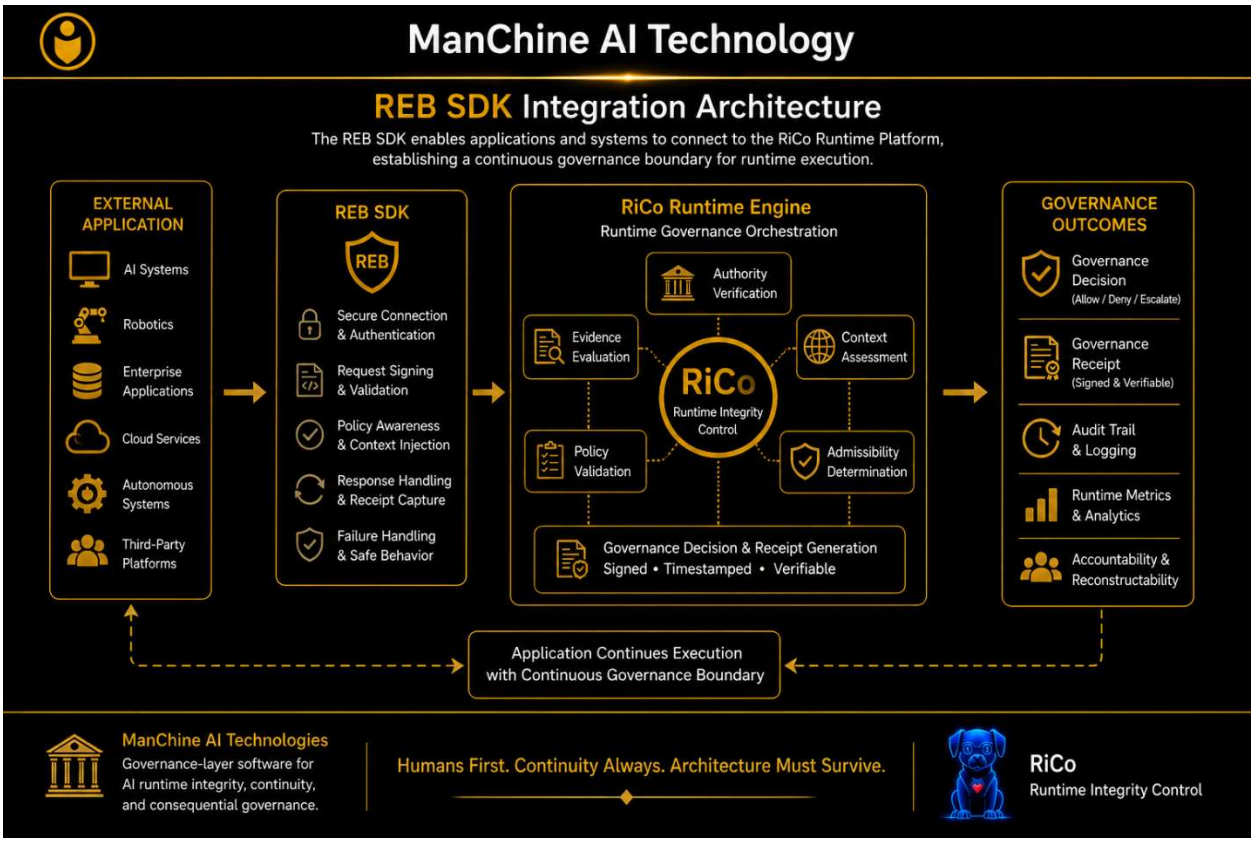
Potential integration targets include:

- Enterprise Resource Planning (ERP) systems
- Customer Relationship Management (CRM) platforms
- Autonomous AI agents
- Multi-agent systems
- Robotics
- Healthcare applications
- Financial systems
- Manufacturing platforms
- Government systems
- Distributed cloud services
- Future intelligent execution environments

This interoperability enables Runtime Integrity Control to function as a governance layer across diverse operational domains.

5.3 Runtime Integration Lifecycle

The REB SDK introduces a standardized governance checkpoint into the execution lifecycle.



The SDK does not determine legitimacy.

It transports governance requests, receives governance decisions, and ensures that execution proceeds in accordance with the Runtime Engine's determination.

5.4 Architectural Responsibilities

The REB SDK performs several responsibilities during runtime integration.

These include:

- Detecting consequential execution points.
- Packaging governance evaluation requests.
- Transmitting governance context to the Runtime Engine.
- Receiving governance determinations.

- Returning governance outcomes to the executing system.
- Preserving execution traceability.
- Supporting reconstructable governance workflows.

Importantly, the SDK does not contain governance policy.

It contains governance integration.

Policies remain managed by the Runtime Platform itself.

5.5 Design Principles

The REB SDK is designed according to several architectural principles.

Platform Independence

The SDK remains independent of programming language, operating system, AI model, or application framework.

Minimal Intrusion

Organizations should integrate governance without replacing existing execution logic.

Deterministic Integration

Every governance request follows a consistent execution pathway.

Interoperability

Applications using different technologies can participate within the same Runtime Integrity Control environment.

Extensibility

Future governance capabilities may be incorporated without requiring fundamental redesign of integrated systems.

5.6 From Architecture to Integration

The Runtime Execution Boundary was introduced in White Paper 002 as the architectural location where legitimacy is continuously evaluated.

The REB SDK transforms that architectural principle into an operational integration capability.

Rather than remaining a conceptual boundary, the Runtime Execution Boundary becomes a standardized software interface capable of connecting intelligent systems to continuous runtime governance.

In this way, the REB SDK represents the practical realization of one of the foundational architectural concepts established throughout the RiCo white paper series.

General's Thought

I think one sentence in this section is worth remembering because it captures the role of the SDK perfectly:

"The REB SDK does not contain governance policy. It contains governance integration."

6. Runtime Governance API

The **Runtime Governance API** provides the standardized service interface through which external systems interact with the RiCo Runtime Platform.

While the Runtime Execution Boundary (REB) SDK enables direct software integration at consequential execution points, the Runtime Governance API exposes governance capabilities as interoperable runtime services.

This allows applications, enterprise platforms, autonomous agents, cloud services, and distributed execution environments to request governance assessments without requiring direct access to the internal operation of the RiCo Runtime Engine.

The Runtime Governance API transforms Runtime Integrity Control from a local governance capability into a platform service that may be consumed across heterogeneous computing environments.

6.1 Governance as a Service

Modern intelligent systems increasingly operate across multiple platforms, cloud environments, organizations, and autonomous services.

Governance must therefore become equally accessible.

Rather than embedding governance logic independently within every application, organizations can expose governance through standardized service interfaces.

The Runtime Governance API enables this architectural approach.

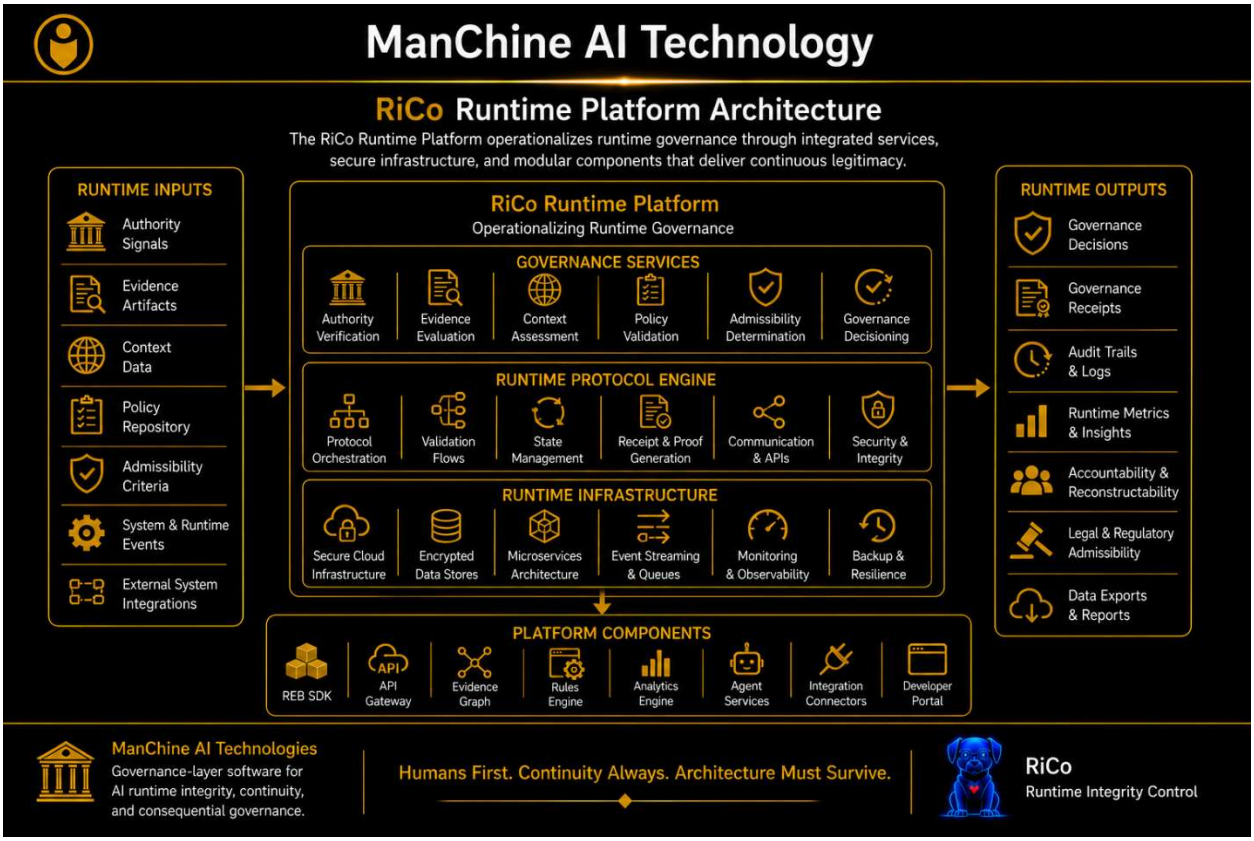
Execution systems remain responsible for operational behavior.

The Runtime Governance API provides access to the governance services required to determine whether proposed execution remains legitimate.

In this way, Runtime Integrity Control functions as **Governance as a Service (GaaS)**.

6.2 Runtime Service Architecture

The Runtime Governance API operates as a shared governance interface between executing systems and the RiCo Runtime Platform.



The API abstracts the complexity of runtime governance while preserving consistent governance behavior across diverse execution environments.

6.3 Core Governance Services

The Runtime Governance API provides standardized access to governance capabilities throughout the execution lifecycle.

Representative services include:

- Submit Governance Evaluation Request
- Request Admissibility Assessment
- Retrieve Governance Decision

- Query Authority Status
- Retrieve Applicable Policy
- Request Evidence Validation
- Obtain Governance Receipt
- Query Runtime Governance History
- Monitor Governance Events
- Retrieve Runtime Trust Status

These services represent logical governance capabilities rather than implementation-specific interfaces.

Future implementations may expose these capabilities through REST, GraphQL, gRPC, event-driven messaging, or other interoperable communication protocols.

6.4 Stateless Governance Access

The Runtime Governance API is designed to remain independent of any specific application or execution environment.

Each governance request carries the information necessary to evaluate legitimacy at the Runtime Execution Boundary.

This architectural approach enables:

- Platform independence
- Horizontal scalability
- Distributed governance
- Multi-tenant deployment
- Hybrid cloud operation
- Federated governance services

The API therefore supports governance wherever execution occurs.

6.5 Governance Response Model

Each governance request produces a structured response describing the current governance assessment.

Typical response information may include:

- Governance Decision
- Admissibility Status
- Supporting Policy References
- Authority Verification
- Evidence Completeness
- Runtime Context Summary
- Governance Receipt Identifier
- Recommended Follow-up Actions

The Runtime Governance API does not simply approve or reject execution.

It explains the governance basis upon which that determination was reached.

This transparency supports accountability, reconstructability, and operational trust.

6.6 Architectural Independence

The Runtime Governance API does not depend upon:

- a specific artificial intelligence model,
- a particular enterprise application,
- a single programming language,
- one cloud provider,
- or one orchestration framework.

Instead, it defines a common governance interface capable of supporting diverse execution environments while preserving consistent runtime governance behavior.

This architectural independence enables Runtime Integrity Control to function as shared governance infrastructure rather than application-specific software.

6.7 From Integration to Governance Services

The Runtime Execution Boundary (REB) SDK establishes **where** governance enters execution.

The Runtime Governance API defines **how** governance capabilities become available to participating systems.

Together they transform Runtime Integrity Control into an interoperable runtime platform capable of supporting governance across heterogeneous intelligent systems.

Rather than embedding governance independently within every application, organizations can rely upon a common governance service that continuously evaluates legitimacy wherever consequential execution occurs.

7. Governance Protocol SDK

The **Governance Protocol SDK** provides the implementation framework for the Runtime Protocol introduced in **White Paper 004**.

While the Runtime Governance API exposes governance capabilities as runtime services, the Governance Protocol SDK establishes how governance information is exchanged consistently between participating systems.

Its purpose is not to define governance policy.

Its purpose is to standardize governance communication.

Through the Governance Protocol SDK, independently developed systems can exchange governance information while preserving consistency, reconstructability, and interoperability across heterogeneous execution environments.

7.1 From Protocol to Implementation

White Paper 004 introduced the Runtime Protocol as the continuous operational process through which governance information accompanies consequential execution.

The Governance Protocol SDK operationalizes that protocol.

Rather than describing governance interactions conceptually, the SDK provides a standardized implementation model through which runtime systems exchange governance information in a consistent and interoperable manner.

The protocol therefore becomes executable.

7.2 Governance Communication

Every consequential execution produces governance information.

That information must remain understandable beyond the boundaries of a single application or organization.

The Governance Protocol SDK enables participating systems to exchange governance artifacts including:

- Governance Requests
- Authority Assertions
- Policy References
- Evidence References
- Operational Context
- Admissibility Assessments
- Governance Decisions
- Governance Receipts
- Runtime Events

These exchanges preserve a common governance vocabulary regardless of the technologies used by participating systems.

7.3 Interoperable Runtime Governance

Modern intelligent systems rarely operate in isolation.

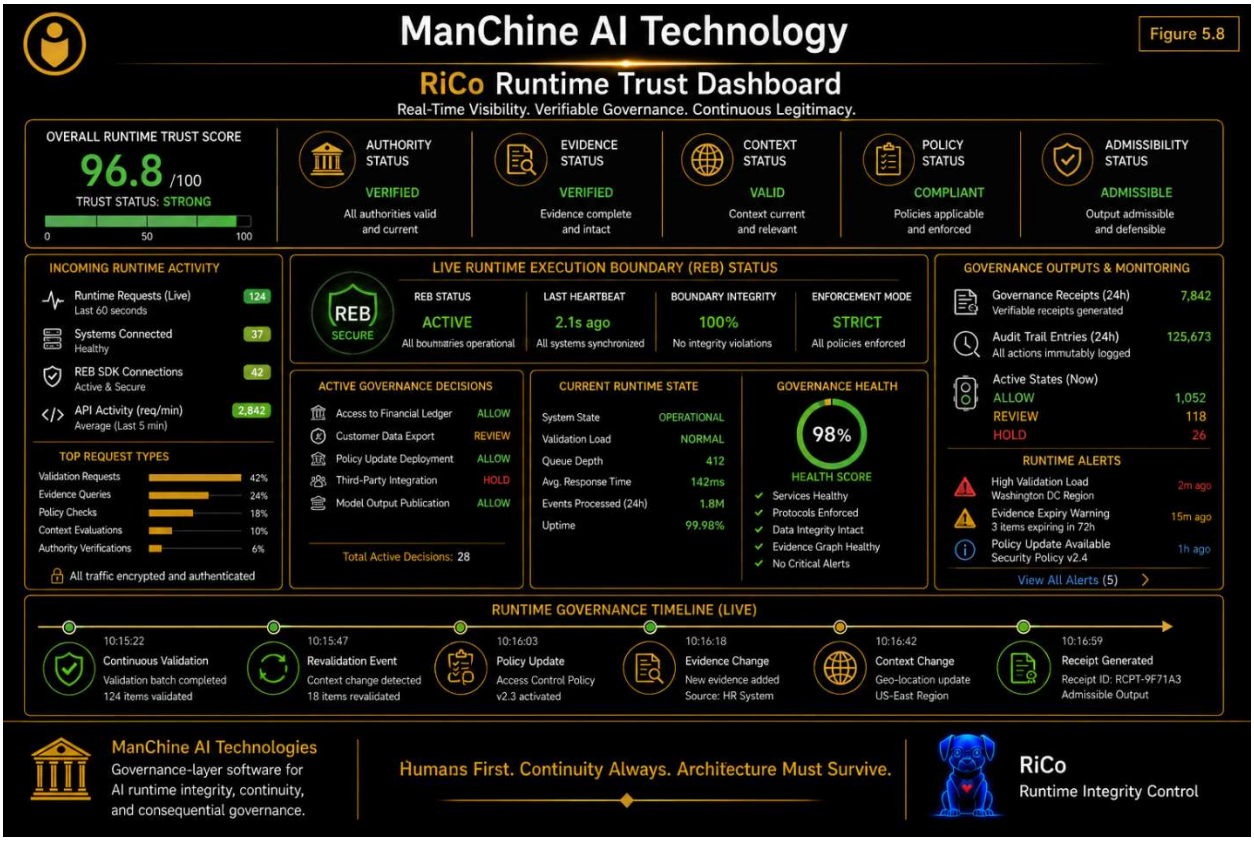
Execution frequently spans:

- enterprise applications
- AI agents
- cloud platforms
- autonomous services
- distributed workflows
- external organizations

Governance must therefore remain interoperable across organizational and technological boundaries.

The Governance Protocol SDK provides the common protocol through which these environments communicate governance information while preserving accountability and reconstructability.

7.4 Runtime Protocol Flow



The protocol does not perform governance evaluation.

It transports governance information in a standardized, verifiable, and interoperable manner.

7.5 Design Principles

The Governance Protocol SDK is guided by several architectural principles.

Protocol Independence

The governance model remains independent of transport technologies.

Whether systems communicate through REST, GraphQL, gRPC, event streams, message brokers, or future communication mechanisms, the governance semantics remain consistent.

Interoperability

Organizations should not redesign existing software to participate in runtime governance.

The protocol enables heterogeneous systems to exchange governance information while preserving architectural independence.

Traceability

Every governance exchange contributes to a reconstructable governance history.

Governance information remains attributable, verifiable, and reviewable throughout the execution lifecycle.

Extensibility

As governance requirements evolve, new protocol elements may be introduced without disrupting existing implementations.

The protocol is designed to evolve while preserving backward compatibility.

7.6 Toward Governance Standards

The Governance Protocol SDK represents more than a software development toolkit.

It establishes the foundation for interoperable runtime governance.

Today, organizations often ask:

Can our application integrate with this platform?

As runtime governance matures, a different question is likely to emerge:

Can our governance platform communicate using the RiCo Runtime Protocol?

This represents a shift from application integration toward governance interoperability.

The objective is not to create proprietary governance.

The objective is to enable independent systems to exchange governance information through a shared architectural language while preserving local authority, policy, and operational autonomy.

7.7 From Platform to Ecosystem

The RiCo Runtime Platform provides deployable governance services.

The Governance Protocol SDK extends those services beyond individual deployments by enabling governance-aware systems to communicate through a common protocol.

In doing so, Runtime Integrity Control evolves from a platform architecture toward a governance ecosystem capable of supporting interoperable runtime governance across organizations, industries, and future intelligent infrastructures.

8. Evidence Graph

The **Evidence Graph** preserves the governance lineage of consequential execution throughout the runtime lifecycle.

While the Runtime Engine evaluates legitimacy and the Runtime Protocol coordinates governance communication, the Evidence Graph maintains the relationships between the governance artifacts that support every consequential decision.

Rather than storing isolated audit records, the Evidence Graph represents governance as an interconnected structure whose relationships remain reconstructable over time.

Every consequential execution therefore carries not only an outcome, but also a traceable chain of governance evidence explaining why that outcome was considered legitimate when it occurred.

8.1 Beyond Traditional Audit Logs

Conventional audit systems typically record events as chronological entries.

While useful for historical review, sequential logs often make it difficult to reconstruct the complete governance basis supporting a consequential decision.

The Evidence Graph approaches governance differently.

Instead of preserving isolated events, it preserves the relationships among governance artifacts.

These relationships may include:

- Authority
- Evidence
- Policy
- Operational Context
- Governance Assessments
- Admissibility Determinations
- Governance Receipts
- Consequential Outcomes

The resulting governance structure enables every consequential action to be reconstructed from its originating conditions through its final execution.

8.2 Governance Lineage

Every governance decision emerges from relationships established during runtime.

A governance determination is not simply an isolated approval or denial.

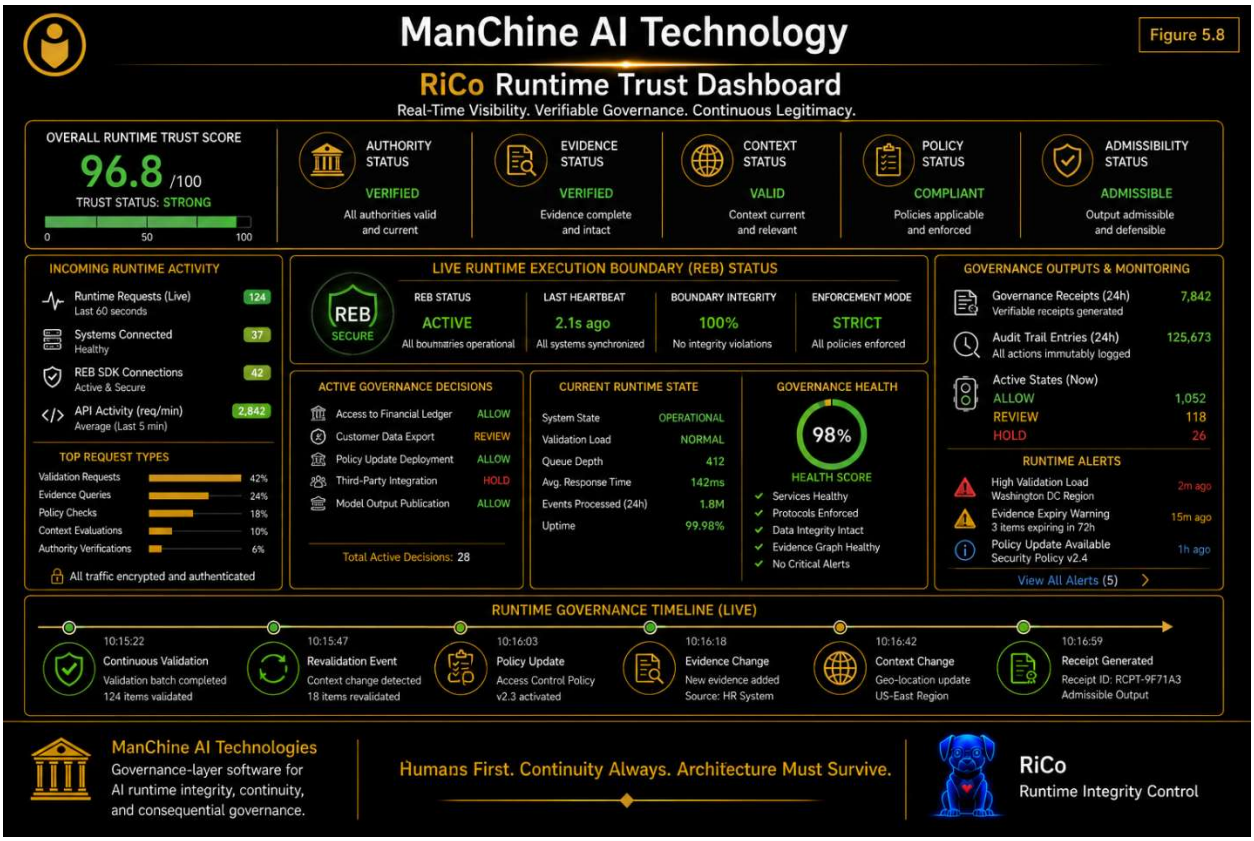
It represents the outcome of interconnected governance information evaluated at a specific moment in time.

The Evidence Graph preserves these relationships by maintaining lineage between:

- the authority responsible for governance,
- the evidence supporting the request,
- the applicable governance policies,
- the operational context,
- the admissibility assessment,
- the governance determination,
- and the resulting consequence.

This lineage allows governance to remain transparent, explainable, and reconstructable even as runtime conditions evolve.

8.3 Conceptual Evidence Graph



Rather than representing a strict execution sequence, the Evidence Graph illustrates the governance relationships that collectively justify consequential execution.

Each node contributes to the governance basis preserved throughout runtime.

8.4 Reconstructability

One of the principal objectives of Runtime Integrity Control is reconstructability.

At any point following consequential execution, authorized reviewers should be capable of determining:

- what execution occurred,
- which authority permitted it,

- which policies governed it,
- what evidence supported it,
- what operational context existed,
- how admissibility was determined,
- and why the resulting consequence was considered legitimate.

The Evidence Graph preserves these governance relationships without requiring investigators to infer them retrospectively.

Governance therefore remains explainable by design rather than reconstructed through interpretation.

8.5 Governance Evolution

Governance is rarely static.

Policies change.

Authority changes.

Evidence evolves.

Operational environments change.

Rather than overwriting previous governance relationships, the Evidence Graph preserves governance history across time.

Successive governance assessments therefore extend the graph instead of replacing it.

This enables organizations to understand not only **what** governance determination was reached, but **how** governance evolved throughout the lifecycle of consequential execution.

8.6 Architectural Independence

The Evidence Graph remains independent of any specific storage technology, database implementation, graph engine, or software platform.

Its purpose is architectural rather than implementation-specific.

Organizations may implement governance lineage using graph databases, relational systems, document stores, distributed ledgers, or future persistence technologies while preserving the conceptual relationships defined by Runtime Integrity Control.

This separation ensures that governance architecture remains stable even as implementation technologies evolve.

8.7 From Governance Records to Governance Knowledge

Traditional audit systems preserve records.

The Evidence Graph preserves relationships.

This distinction transforms governance from a collection of historical events into an interconnected body of governance knowledge capable of supporting continuous explanation, accountability, and legitimate consequential execution.

Within the RiCo Runtime Platform, the Evidence Graph therefore serves as the architectural memory of runtime governance, preserving not merely **what** occurred, but **why** those consequences were considered legitimate when they occurred.

9. Runtime Trust Dashboard

The **Runtime Trust Dashboard** provides live visibility into governance activity across the RiCo Runtime Platform.

While the RiCo Runtime Engine evaluates legitimacy, the Runtime Execution Boundary (REB) governs consequential transitions, and the Evidence Graph preserves governance lineage, the Runtime Trust Dashboard makes those processes visible to authorized human reviewers.

Its purpose is not to replace governance judgment.

Its purpose is to make runtime governance observable.

The dashboard presents the current state of authority, evidence, policy, context, admissibility, and consequential execution in a form that supports oversight, intervention, accountability, and review.

9.1 From System Monitoring to Governance Visibility

Traditional operational dashboards focus on system health.

They display metrics such as:

- availability,
- performance,

- latency,
- throughput,
- failures,
- and resource utilization.

These metrics help determine whether a system is functioning.

They do not necessarily determine whether its continued execution remains legitimate.

The Runtime Trust Dashboard addresses this gap by displaying governance conditions alongside operational state.

It enables authorized users to distinguish between two different questions:

Is the system continuing to operate?

and

Does its continued execution remain legitimately supported?

This distinction reflects the separation between operational continuity and legitimacy continuity established in White Paper 002.

3M

9.2 Governance State Visualization

The Runtime Trust Dashboard may display governance information including:

- Current admissibility status
- Active Runtime Execution Boundaries
- Authority validity
- Evidence completeness
- Applicable policy versions
- Context changes
- Pending reviews
- Active holds
- Denied execution requests
- Governance receipt status
- Runtime revalidation events
- Consequence-bearing execution activity

The dashboard therefore presents governance as a live operational state rather than a static compliance record.

3M

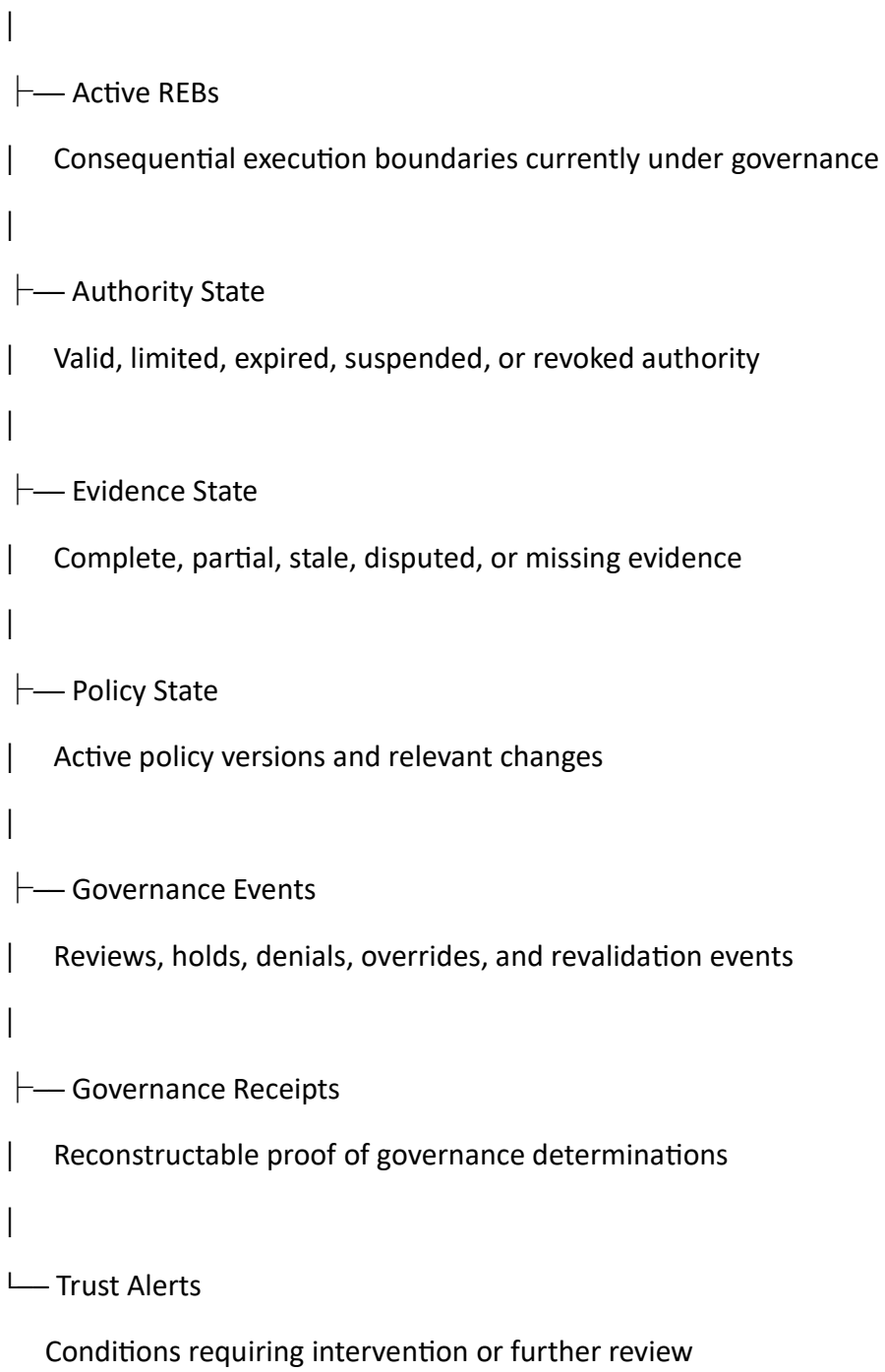
9.3 Conceptual Dashboard Structure

Runtime Trust Dashboard

|

├— Runtime Legitimacy

| Current admissibility and revalidation status



The dashboard does not produce governance determinations independently.

It displays the governance state produced by the RiCo Runtime Platform.

9.4 Runtime Trust Is Not a Single Score

The Runtime Trust Dashboard should not reduce governance to one universal trust score.

A single number may conceal important differences among:

- authority,
- evidence,
- policy,
- context,
- admissibility,
- and consequence.

For this reason, runtime trust is represented as a multidimensional governance condition rather than a simplified rating.

A system may have valid authority but incomplete evidence.

It may satisfy policy while operating under changed context.

It may remain operationally healthy while legitimacy is degrading.

The dashboard must preserve these distinctions instead of collapsing them into a single indicator.

9.5 Human Reachability and Intervention

The dashboard preserves meaningful human reachability during consequential execution.

Authorized reviewers may use it to:

- inspect the basis of a governance determination,
- identify missing or stale evidence,
- review authority changes,
- examine active holds,
- evaluate policy conflicts,
- request additional review,
- redirect execution,
- or withdraw legitimacy when required.

The objective is not constant human micromanagement.

The objective is to ensure that human responsibility remains capable of reaching execution before consequence becomes irreversible.

9.6 Alerts and Governance-Relevant Change

Not every operational change requires governance intervention.

The Runtime Trust Dashboard therefore emphasizes changes that may materially affect legitimacy.

Examples include:

- authority expiration or revocation,
- policy amendment,
- evidence degradation,
- conflicting observations,
- contextual change,
- scope expansion,
- unresolved review requirements,
- or divergence between operational and legitimacy continuity.

These events may trigger targeted revalidation by the RiCo Runtime Engine and corresponding dashboard alerts.

The dashboard therefore supports event-driven governance rather than indiscriminate monitoring.

9.7 Reconstructable Oversight

Every displayed governance event should remain linked to its supporting evidence and governance receipt.

An authorized reviewer should be able to move from:

alert

to:

governance assessment

to:

authority, evidence, policy, and context

to:

resulting consequence.

This linkage connects the Runtime Trust Dashboard directly to the Evidence Graph.

The dashboard provides visibility.

The Evidence Graph provides reconstructability.

Together they enable governance oversight that is both immediate and explainable.

9.8 Role-Based Governance Views

Different users require different governance views.

A technical operator may need to see active REBs and runtime alerts.

A governance officer may need to inspect authority, policy, and admissibility.

An auditor may need access to governance receipts and historical lineage.

An executive may require a high-level view of consequential risk and unresolved governance conditions.

The Runtime Trust Dashboard should therefore support role-aware access while preserving the integrity of the underlying governance record.

Visibility may vary by responsibility.

Governance truth must not.

9.9 From Visibility to Operational Trust

The Runtime Trust Dashboard transforms runtime governance into an observable organizational capability.

It allows institutions to see not only whether intelligent systems are functioning, but whether the conditions supporting consequential execution remain valid.

Within the RiCo Runtime Platform, trust is therefore not presented as a static declaration.

It is presented as a continuously inspectable governance state supported by authority, evidence, policy, context, admissibility, and reconstructable consequence.

Operational dashboards show whether systems are running.
The Runtime Trust Dashboard shows whether their consequences remain legitimate.

10. Future Direction

The RiCo Runtime Platform represents the operational realization of the governance architecture established throughout the first four papers in this series.

It is not presented as the final stage of runtime governance.

Rather, it establishes a foundation upon which future governance capabilities may be developed.

As intelligent systems continue to evolve, governance challenges will increasingly extend beyond individual applications, organizations, and execution environments.

Autonomous agents will collaborate across organizational boundaries.

Enterprise platforms will exchange consequential decisions through distributed workflows.

Public and private infrastructures will increasingly depend upon intelligent systems whose decisions affect people, institutions, and critical operations.

These environments will require governance capabilities that are not only operational, but also interoperable.

10.1 From Platform to Ecosystem

The RiCo Runtime Platform is intentionally designed as modular governance infrastructure.

Its components may be deployed independently or together while preserving a common governance architecture.

This modularity creates opportunities for future interoperability among governance-aware systems developed by different organizations and operating across different technologies.

Future work may explore standardized governance exchanges that enable runtime governance to extend beyond a single implementation while preserving local authority, policy, and organizational autonomy.

10.2 Toward Interoperable Governance

Historically, software interoperability has focused on exchanging data.

Runtime governance introduces an additional challenge.

Systems must increasingly exchange the governance basis upon which consequential execution is permitted.

This includes communicating:

- authority,
- evidence,
- policy,
- operational context,
- admissibility,
- governance determinations,
- and reconstructable governance receipts.

The long-term objective is not centralized governance.

The objective is interoperable governance.

10.3 Future Platform Capabilities

The RiCo Runtime Platform provides a conceptual foundation for future software capabilities that may include:

- RiCo Runtime Engine
- Runtime Execution Boundary (REB) SDK
- Runtime Governance API
- Governance Protocol SDK
- Evidence Graph
- Runtime Trust Dashboard

Additional capabilities may emerge as runtime governance matures and new operational requirements are identified through research, implementation, and collaboration.

The architecture is therefore intended to evolve through practical experience rather than remain fixed to a single implementation.

10.4 Standards Through Collaboration

The concepts presented throughout this white paper series are intended to contribute to the broader discussion surrounding AI governance.

Progress in governance will not result from isolated architectures operating independently.

It will emerge through comparison, experimentation, constructive critique, and collaboration among researchers, developers, standards organizations, industry, and public institutions.

Future interoperability should therefore be guided by shared architectural principles rather than proprietary isolation.

10.5 The Next Question

The first four white papers asked foundational questions.

What is Runtime Integrity Control?

Where should governance occur?

What governance components participate?

How do those components operate together?

This paper has asked a new question:

How can runtime governance become deployable software?

The questions that follow are likely to extend beyond individual implementations.

They may ask how independent governance systems communicate, cooperate, and preserve legitimacy across increasingly interconnected intelligent environments.

Those questions remain open.

They also represent an opportunity for continued research.

Closing Perspective

The RiCo Runtime Platform is presented as one possible architectural direction for operationalizing Runtime Integrity Control.

Its purpose is not to replace existing artificial intelligence systems, enterprise platforms, or governance frameworks.

Its purpose is to complement them by providing a governance layer capable of continuously evaluating consequential execution as authority, evidence, context, policy, and admissibility evolve throughout runtime.

Whether this architecture ultimately influences future governance practices will depend not upon claims of novelty, but upon continued evaluation, implementation, comparison, and constructive scientific dialogue.

The discussion remains open.

11. Conclusion

The development of Runtime Integrity Control (RiCo) has followed a deliberate architectural progression.

Each paper in this series has addressed a distinct aspect of runtime governance.

White Paper 001 introduced Runtime Integrity Control as a governance architecture for preserving the legitimacy of consequential execution.

White Paper 002 established the Runtime Execution Boundary (REB) and distinguished operational continuity from legitimacy continuity, demonstrating that continued execution alone is insufficient to justify continued consequence.

White Paper 003 defined the Runtime Governance Stack, identifying the governance components required to preserve legitimacy throughout runtime.

White Paper 004 introduced the Runtime Protocol, describing how those components cooperate through continuous validation, revalidation, accountability, and reconstructable governance.

This paper has taken the next logical step.

Rather than introducing additional governance concepts, it has presented the **RiCo Runtime Platform** as the operational realization of the architecture established throughout the previous papers.

The platform demonstrates how runtime governance can be organized as interoperable software services capable of supporting continuous legitimacy evaluation across heterogeneous intelligent systems.

Its components—including the RiCo Runtime Engine, the Runtime Execution Boundary (REB) SDK, the Runtime Governance API, the Governance Protocol SDK, the Evidence Graph, and the Runtime Trust Dashboard—represent a modular approach to operationalizing governance while preserving the architectural separation between execution and legitimacy.

Throughout this series, one principle has remained consistent.

RiCo is not intended to replace existing artificial intelligence systems.

It is designed to complement them by providing a governance layer that continuously evaluates whether consequential execution remains legitimate as authority, evidence, context, policy, and admissibility evolve throughout runtime.

The continued advancement of AI governance will require more than increasingly capable models.

It will require governance infrastructures capable of operating continuously, remaining reconstructable, supporting interoperability, and preserving legitimate consequence across changing operational conditions.

The RiCo Runtime Platform is presented as one architectural contribution toward that objective.

Its value will ultimately be determined not by its description within these pages, but by continued research, constructive critique, practical implementation, and open scientific dialogue.

The work presented throughout this white paper series is therefore offered as a contribution to that ongoing discussion.

The conversation continues.

Humans First. Continuity Always. Architecture Must Survive.

RiCo governs consequential execution across foundation models, orchestration frameworks, agent platforms, and enterprise software rather than replacing them.

The architecture remains independent of the underlying AI implementation.

Why RiCo Complements Existing AI

The RiCo Runtime Platform represents the operational realization of the Runtime Governance Stack.

As intelligent systems continue to evolve, governance will increasingly require interoperable runtime services rather than isolated policy documents.

The long-term direction is not simply software deployment. It is interoperability.

Eventually the question will shift from: "Can RiCo govern this system?" to: "Can this system speak the RiCo Runtime Protocol?"

Looking Ahead

References

RiCo White Paper Series

Colimon, R. (2026). *White Paper 001: Runtime Integrity Control (RiCo).* ManChine AI Technology LLC.

Colimon, R. (2026). *White Paper 002: Operational Continuity vs. Legitimacy Continuity — The Runtime Execution Boundary (REB).* ManChine AI Technology LLC.

Colimon, R. (2026). *White Paper 003: The Runtime Governance Stack.* ManChine AI Technology LLC.

Colimon, R. (2026). *White Paper 004: The Runtime Protocol.* ManChine AI Technology LLC.

Governance and AI References

National Institute of Standards and Technology (NIST). *AI Risk Management Framework (AI RMF 1.0).* U.S. Department of Commerce.

ISO/IEC 42001. *Artificial Intelligence — Management System.*

OECD. *OECD Principles on Artificial Intelligence.*

European Union. *Artificial Intelligence Act.*

Systems and Architecture

Saltzer, J. H., Reed, D. P., & Clark, D. D. (1984). *End-to-End Arguments in System Design.*

Lamport, L. *Time, Clocks, and the Ordering of Events in a Distributed System.*

National Institute of Standards and Technology (NIST). *Zero Trust Architecture (SP 800-207).*

Runtime and Distributed Systems

Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures.*

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software.*

Future Reading

Additional references will be incorporated as the RiCo Runtime Platform evolves through implementation, interoperability research, and continued collaboration with the AI governance community.