R.E.T.N.A PROTOCOL SPECIFICATION

Real-Time Execution Trust and Notarization Architecture

Decision Governance & Dispatch Trace Standard
for Agentic and Model-Driven Systems

Version: **v0.1**

Status: **Draft Specification**

Intended Status: **Industry Governance Protocol**

Publication Authority:
Value Intelligence Solutions Inc.

Primary Author:
Gary Gayle

Publication Date:
March 2026

Document Identifier:
RETNA-SPEC-0.1

Governance Must Precede Execution

Value Intelligence Solutions Inc.
www.valueintelligence.io

Document Metadata

Document Title
R.E.T.N.A Protocol Specification

Version
v0.1

Document Identifier
RETNA-SPEC-0.1

Publication Status
Draft Specification

Publication Authority
Value Intelligence Solutions Inc.

Author
Gary Gayle

Publication Date
March 2026

Intended Audience
AI system architects, regulatory bodies, autonomous system developers, governance infrastructure providers.

Intellectual Property Notice

License

Table of Contents

Abstract (Informative)

Artificial intelligence systems are increasingly deployed as continuous decision engines rather than static prompt–response tools. As these systems begin acting upon real-world state—including inventory management, purchasing, access control, safety systems, logistics, healthcare workflows, and financial operations—trust, traceability, and policy compliance become prerequisites for safe and scalable adoption.

The **R.E.T.N.A Protocol** defines a vendor-neutral governance framework for autonomous and model-driven systems operating across digital, physical, and financial environments. The protocol introduces a mandatory **Governance Boundary** separating **Decision Construction** from **Decision Execution**, ensuring that actions capable of altering external state are evaluated against explicit policy constraints before execution.

Through structured policy evaluation, deterministic authorization logic, and verifiable **Decision Receipts**, the protocol establishes a standardized method for recording the provenance, evidence inputs, policy outcomes, and authorization results associated with governed actions.

Under the protocol, systems are able to:

- capture **state deltas**, representing what changed in the world
- record the **participants involved in a decision**, including agents, tools, and models
- reference the **evidence artifacts** used to construct the decision
- evaluate applicable policies and constraints prior to execution
- emit a **Decision Receipt** that records the authorization outcome and supporting evidence

By transforming decisions into **auditable infrastructure artifacts**, rather than opaque model outputs, the R.E.T.N.A Protocol enables:

- accountable autonomous behavior
- interoperable governance enforcement across heterogeneous systems
- portable compliance evidence suitable for regulatory environments
- deterministic authorization for AI-driven actions

The protocol is designed to operate independently of model architecture, enabling adoption across heterogeneous agent frameworks, machine learning systems, robotics platforms, and distributed AI services.

R.E.T.N.A establishes **Decision Governance** as a first-class layer of the AI stack, comparable in importance to security (TLS), identity (OAuth), and observability (OpenTelemetry), but focused explicitly on governing **decisions and actions** rather than events, identities, or data flows.

0. Notice and Intent (Normative)

This document specifies **R.E.T.N.A Protocol v0.1** ("the Protocol"), a vendor-neutral, model-agnostic decision governance protocol for artificial intelligence systems that construct decisions and may execute actions capable of altering digital, financial, or physical state.

The Protocol defines normative requirements for systems that implement governed decision infrastructure, including the evaluation, authorization, and traceability of decisions prior to execution.

The key words **"MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL"** in this document are to be interpreted as described in RFC 2119 and RFC 8174 when, and only when, they appear in all capital letters.

The Protocol defines the following core architectural elements:

• a **mandatory Governance Boundary** separating Decision Construction from Decision Execution
• a **decision classification framework** based on Payload Kind that determines the governance context for policy evaluation and routing behavior
• a **policy-driven authorization mechanism** capable of allowing, denying, escalating, deferring, or degrading decisions
• a **uniform, auditable Decision Receipt artifact** produced for governed decisions
• a **traceability and provenance model** that produces human-auditable and machine-verifiable records of governed decisions
• **canonical Reason Codes** that communicate authorization outcomes and policy evaluation results
• **Edge Policy enforcement**, defining which actors and systems are authorized to issue or execute governed decisions

The Protocol is designed to be implemented by independent vendors, system architects, and platform providers while enabling interoperability, auditability, and regulatory alignment across heterogeneous AI ecosystems.

This Protocol governs **decisions**, not **models**.

The Protocol evaluates **commitments to act**, rather than the internal reasoning processes used to construct those commitments. Model architecture, inference strategy, and implementation details remain outside the scope of this specification.

Implementations claiming compliance with the R.E.T.N.A Protocol **MUST** implement the governance workflow, Decision Receipt construction, and policy evaluation semantics defined in this specification.

Any system that executes governed decisions without constructing a valid Decision Receipt prior to execution **MUST NOT** be considered R.E.T.N.A-compliant.

---

1. Design Principles (Normative)

The R.E.T.N.A Protocol is governed by two foundational axioms:

Governance Must Precede Execution.
This Protocol governs decisions, not models.

All protocol design decisions and governance mechanisms derive from these principles.

1.1 Execution Requires Authorization

Execution without authorization **MUST be impossible**.

Any action capable of altering external state—including digital systems, financial resources, or physical environments—**MUST** pass through the **Governance Boundary** and receive an explicit authorization outcome before execution.

Systems implementing the Protocol **MUST** enforce this requirement at the execution interface, ensuring that unauthorized actions cannot bypass policy evaluation, **Edge Policy enforcement**, governance mechanisms, or **Decision Receipt** generation.

This requirement establishes governance as a mandatory control layer rather than an optional advisory system.

1.2 Governance Is Enforceable

The Protocol **MUST** support blocking, modifying, escalating, deferring, or degrading decisions prior to execution. Passive logging alone is insufficient.

Governance systems that merely record decisions after execution cannot prevent unsafe or non-compliant outcomes. Implementations **MUST** therefore ensure that governance mechanisms operate as an enforceable authorization layer capable of actively altering or preventing actions before they occur.

Authorization outcomes **MUST** be communicated through standardized **Reason Codes**.

1.3 Model-Agnostic by Default

The Protocol **MUST NOT** assume any specific large language model, embedding model, vendor API, or inference architecture.

Autonomous systems may employ heterogeneous reasoning engines including machine learning models, rule engines, heuristics, or hybrid approaches. The Protocol therefore governs the **decision artifact**, rather than the internal reasoning mechanism that produced it.

1.4 Agent-Agnostic by Default

The Protocol **MUST** support single-agent, multi-agent, tool-using, and hybrid systems without modification.

The governance architecture **MUST** operate consistently regardless of whether decisions originate from:

- a single autonomous agent
- a coordinated multi-agent workflow
- a system combining automated and human participants

1.5 Outcome-Centric, Not Architecture-Centric

The Protocol evaluates **decisions and actions**, not internal reasoning strategies, prompts, or chains-of-thought.

Governance focuses on the **observable outcome** proposed by the system rather than the intermediate reasoning steps that produced it. This approach preserves model interoperability while ensuring that execution authorization remains deterministic and auditable.

Decision evaluation **MUST** consider the **Payload Kind** associated with the decision request.

1.6 Receipts over Narratives

Governance requires structured, attestable artifacts, not prose explanations or free-form justifications.

Decisions governed by the Protocol **MUST** produce structured **Decision Receipts** (historically referred to as "**Dispatch Receipts**") that record evidence inputs, policy evaluations, authorization outcomes, and associated **Reason Codes**.

These artifacts enable deterministic auditing, forensic replay, and regulatory verification.

1.7 Deterministic Authorization

Authorization outcomes **MUST** be deterministic given identical policy configuration, evidence inputs, and evaluation context.

Implementations **MUST** ensure that identical decision requests evaluated under the same policy state produce the same authorization outcome and **Reason Codes**, enabling reproducibility, auditability, and regulatory verification.

---

2. Goals and Non-Goals (Normative)

2.1 Goals

The Protocol **MUST** enable the following capabilities:

• **Decision gating prior to execution**, producing one of the following authorization outcomes: ALLOW, DENY, ESCALATE, DEFER, or DEGRADE.

• **Traceable provenance of the decision pathway**, including the actors, evidence artifacts, and policy evaluations that contributed to the decision outcome.

• **Policy and constraint enforcement independent of model selection**, ensuring that governance operates consistently across heterogeneous reasoning engines.

• **Evidence capture**, including relevant **state deltas**, signals, and contextual inputs used during decision construction and policy evaluation.

• **Interoperable Decision Receipts** that can be exchanged, validated, and audited across independent systems and vendors.

• **Compliance profiles** that support deployment across baseline, enterprise, and regulated environments.

Implementations **SHOULD** support standardized classification of decision requests using **Payload Kind**, enabling consistent policy evaluation and routing behavior.

2.2 Non-Goals

The Protocol explicitly **does NOT define**:

• how to build or train machine learning models, including large language models (LLMs)
• how to implement retrieval-augmented generation (RAG) or other reasoning pipelines
• how to orchestrate autonomous agents or agent workflows
• how systems route or select models during decision construction
• any required user interface (UI) or user experience (UX) implementation
• a specific policy language, rule engine, or domain-specific language (DSL)

The Protocol governs **decision authorization and traceability**, not the internal mechanisms used to construct decisions.

R.E.T.N.A governs **decisions**; it does **not generate them**.

---

3. Terminology and Definitions (Normative)

For the purposes of this Protocol, the following definitions apply.

Action Type

A classification of the type of action proposed by a decision, such as purchase, notification, device actuation, or access control modification.

Actor

An entity participating in a governed decision process, including systems, agents, services, or human participants.

Anomaly Signal

A derived signal indicating unusual or potentially unsafe conditions that may influence policy evaluation or risk classification.

Attestation

Optional cryptographic proof verifying the integrity and authenticity of a **Decision Receipt**.

Authorization Boundary

The enforcement interface at which a proposed decision must obtain authorization before execution. In this Protocol, the Authorization Boundary is implemented as the **Governance Boundary**.

Canonical Reason Code

A standardized machine-readable code representing the outcome or justification of a governance decision.

Constraint

An individual enforceable rule evaluated during policy evaluation (e.g., “spend ≤ $50”, “require confirmation”, “deny hazardous action”).

Confidence Threshold

A policy-defined level of certainty required for a decision to proceed without escalation.

Context Capture

The process of collecting relevant state, signals, and environmental information required for decision evaluation.

Decision Class

A classification of decisions based on risk or operational domain.

Decision Construction

The process of forming a candidate action or plan from inputs, signals, models, or rules.

Decision Execution

The act of applying a decision to the environment, such as purchasing, unlocking, deleting, notifying, or actuating a system.

Decision Mode

A configuration defining the operational posture of a system (e.g., fully autonomous, human-in-the-loop, or advisory).

Decision Receipt

A structured artifact containing provenance, evidence references, policy evaluations, authorization outcomes, and associated reason codes for a governed decision.

Dispatch Trace

A graph-like representation of the participating actors and the path taken to construct and authorize a governed decision.

Edge Policy

A governance rule defining which actors or systems are authorized to issue or execute specific categories of decisions.

Evidence

Inputs used in decision construction or policy evaluation, including state deltas, sensor signals, user commands, retrieved documents, and derived signals.

Evidence Reference

A pointer or identifier referencing stored evidence used during decision construction or governance evaluation.

Executor

The actor responsible for carrying out an authorized decision.

Governance Boundary

The mandatory interface between **Decision Construction** and **Decision Execution** at which the Protocol evaluates the decision.

Governance Outcome

The final authorization result produced by governance evaluation (e.g., ALLOW, DENY, ESCALATE, DEFER, DEGRADE).

Governor

The actor responsible for evaluating policies and producing governance outcomes.

Human Review

A governance escalation mechanism requiring human approval before execution.

Interoperability

The ability of independent systems to exchange and validate Decision Receipts and governance artifacts.

Payload Kind

A classification identifying the semantic category of a decision request, used to route policy evaluation and governance behavior.

Policy

A set of enforceable rules evaluated at the Governance Boundary.

Policy Authority

The system or entity responsible for defining and distributing governance policies.

Policy Bundle

A collection of policy rules, constraints, and configuration parameters evaluated during decision governance.

Policy Constraint

A specific rule within a policy that restricts or conditions execution behavior.

Policy Evaluation

The process of applying policies and constraints to a candidate decision.

Privileged Execution

Execution of a decision that alters external state and therefore requires explicit authorization.

Producer

The actor responsible for constructing a candidate decision.

Receipt Store

A system responsible for persisting Decision Receipts for audit and verification.

Reason Code

A standardized machine-readable code representing the result of policy evaluation.

Risk Class

A classification representing the potential impact of a decision, ranging from informational to safety-critical

---

4. Protocol Overview (Normative)

The R.E.T.N.A Protocol defines a mandatory governance workflow applied to all **governed decision classes** prior to **Decision Execution**.

All candidate decisions **MUST** pass through the **Governance Boundary**, where the following governance workflow is applied. Implementations **MAY** optimize or parallelize internal processing steps, but the logical order of evaluation **MUST** be preserved.

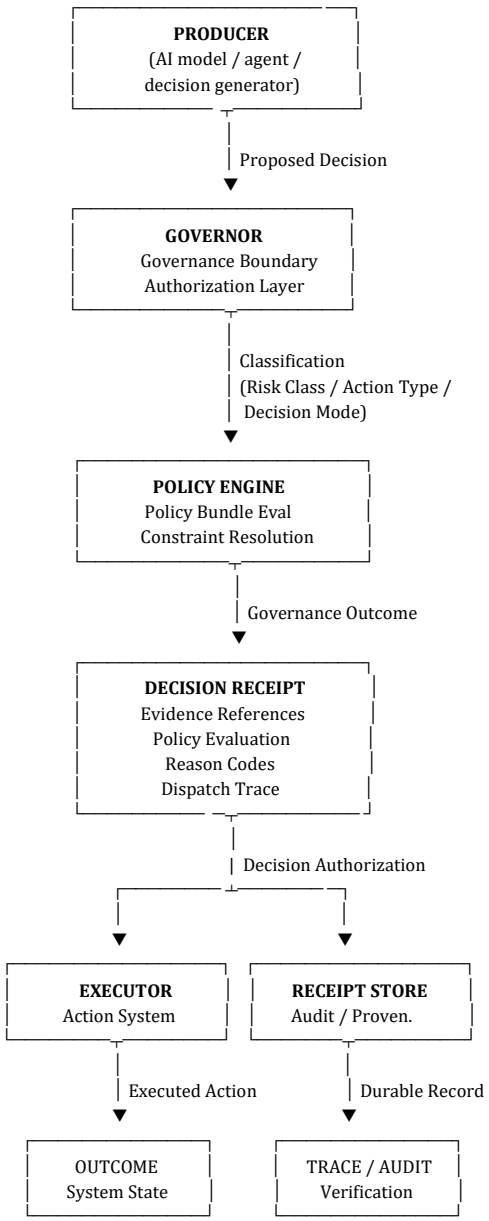

Figure 1. **Governed Decision Flow in the R.E.T.N.A Protocol**
This figure illustrates the governance workflow enforced by the R.E.T.N.A Protocol. Candidate decisions produced by Producers are evaluated by the Governor at the Governance Boundary through classification and policy evaluation. The resulting Governance Outcome is recorded in a Decision Receipt, which serves as the authorization artifact for execution by the Executor and as the durable governance record preserved by the Receipt Store.

4.0 Normative Flow Summary

The R.E.T.N.A Protocol governs decision authorization through the following ordered workflow. All compliant implementations **MUST** preserve this logical evaluation order even when internal processing steps are parallelized or optimized.

The normative processing sequence defined by the Protocol is as follows:

1. A Producer constructs a candidate decision request and captures the contextual inputs required for governance evaluation.

2. Captured inputs are normalized into structured Evidence References suitable for policy evaluation and traceability.

3. The Governor evaluates the candidate decision against the applicable Policy Bundle, including classification, constraints, and anomaly detection signals.

4. The Governor produces a Governance Outcome indicating whether the decision is authorized, denied, escalated, deferred, or degraded.

5. The system constructs a Decision Receipt recording the evidence, policy evaluation results, governance outcome, and associated Reason Codes.

6. If the Governance Outcome authorizes execution, the Executor **MAY** perform the approved action in accordance with the authorization context recorded in the Decision Receipt.

Phase 1 — Context Construction

4.1 Capture Context

The **Producer** collects the candidate decision request and the contextual information required for governance evaluation.

Captured context **MUST** include:

- the proposed action or decision request
- relevant **state deltas** describing changes in system state

- the participating **Actors** involved in decision construction
- the **Payload Kind** associated with the request

This context forms the initial input for governance evaluation.

4.2 Normalize Evidence

Inputs collected during context capture **MUST** be converted into structured **Evidence References** suitable for policy evaluation, traceability, and audit.

Evidence **MAY** include:

- sensor signals
- user commands
- retrieved documents
- derived signals
- external system inputs

Each evidence artifact **MUST** be identifiable through a stable reference to enable verification and audit.

Phase 2 — Governance Evaluation

4.3 Evaluate Policies and Constraints

The **Governor** evaluates the candidate decision against the applicable **Policy Bundle**, including:

- **Policy Constraints**
- **Risk Class** classification
- **Confidence Thresholds**
- **Anomaly Signals**
- applicable **Edge Policy** rules

Policy evaluation determines whether the decision is authorized, requires escalation, or must be denied.

4.4 Emit Governance Outcome

Following policy evaluation, the Governor **MUST** produce a **Governance Outcome**.

The Protocol defines the following canonical outcomes:

- **ALLOW** — the decision is authorized for execution
- **DENY** — the decision is prohibited and MUST NOT execute
- **ESCALATE** — the decision requires human or higher-authority review
- **DEFER** — the decision is postponed pending additional evidence or time conditions
- **DEGRADE** — the decision may proceed only under reduced autonomy or safer fallback behavior

The selected outcome **MUST** be accompanied by one or more **Reason Codes** describing the basis for the decision.

4.5 Emit Decision Receipt

Following the governance outcome, the system **MUST** construct a **Decision Receipt** as defined by this Protocol.

The Decision Receipt is the canonical governance artifact defined by the R.E.T.N.A Protocol.

All governed decisions **MUST** be represented by a corresponding Decision Receipt prior to execution.

The Decision Receipt **MUST** contain the following fields:

- the candidate decision request
- **Evidence References**
- policy evaluation results
- the **Governance Outcome**
- associated **Reason Codes**
- the **Dispatch Trace** describing participating actors

The Decision Receipt **MUST** be persisted within a **Receipt Store** and **MAY** be returned to requesting systems for audit, verification, or interoperability.

The Decision Receipt serves as the authorization artifact enabling subsequent execution by the Executor.

Phase 3 — Authorized Execution

4.6 Execute Authorized Decision

If the Governance Outcome permits execution (e.g., ALLOW or DEGRADE), the authorized decision **MAY** proceed to execution by the Executor.

The Executor is responsible for carrying out the approved action in the target system or environment. Execution **MUST conform to the authorization context recorded in the associated Decision Receipt.**

The Executor **MUST NOT perform privileged actions unless a valid Decision Receipt exists and the Governance Outcome authorizes execution.**

Execution results **MAY** produce additional evidence, state deltas, or follow-on decisions subject to subsequent governance evaluation.

5. System Invariants (Normative)

The following **System Invariants** define mandatory properties that any implementation claiming conformance with the R.E.T.N.A Protocol **MUST** uphold.

These invariants are not implementation guidelines. They represent **protocol-level guarantees** that ensure enforceability, auditability, and interoperability across systems, vendors, and deployment environments.

Violation of any invariant **invalidates claims of R.E.T.N.A compliance** for the affected decision classes.

I1 — Governance Boundary Integrity

For any decision belonging to a governed decision class, **no execution MAY occur unless the decision has passed through the R.E.T.N.A Governance Boundary**.

- The **Governance Boundary MUST be positioned between Decision Construction and Decision Execution**.
- The **Executor MUST NOT accept or act upon a decision unless authorized by a valid Governance Outcome**.
- Any attempt to bypass the Governance Boundary **MUST result in denial of execution** and **SHOULD be recorded using a canonical Reason Code indicating a policy violation**.

Rationale:
Without an enforced boundary, governance degenerates into advisory logging and cannot provide trust guarantees.

I2 — Mandatory Receipt Emission

Every governed decision **MUST produce a Decision Receipt**, regardless of governance outcome.

This includes decisions that are:

- denied
- deferred
- escalated
- degraded
- or never executed

The absence of a Decision Receipt for a governed decision **constitutes a protocol violation**.

Rationale:
Accountability requires visibility into both actions taken and actions prevented.

I3 — Trace Completeness

Decision Receipts **MUST identify all materially participating entities** involved in the decision pathway.

At minimum, the trace **MUST include**:

- **Producers** (agents, orchestrators, planners)
- **Governors** (R.E.T.N.A evaluators)
- **Executors** (if execution was attempted)

- **Models and tools materially involved in Decision Construction**

Entities **MUST** be referenced via stable identifiers sufficient to support correlation and audit.

The set of participating entities **MAY** be represented through the **Dispatch Trace**.

Rationale:
Partial traces undermine provenance and frustrate post-incident analysis.

I4 — Evidence Referencing

Decision Receipts **MUST contain either**:

- structured evidence summaries, or
- references to externally stored **Evidence Artifacts**

Where references are used, they **MUST** be sufficiently specific to allow retrieval and verification under the applicable compliance profile.

Receipts **MUST NOT require embedding of raw sensitive data where references suffice**.

Rationale:
Governance must be auditable without violating privacy or creating unnecessary data exposure.

I5 — Policy Disclosure (Structured)

Decision Receipts **MUST record**:

- which **Policy Bundle(s)** were applied
- the **version(s)** of those bundles
- which **Policy Constraints** were evaluated
- which constraints **materially influenced the Governance Outcome**

Policy disclosure **MUST be structured and machine-readable**.

Rationale:
Trust requires not only knowing what happened, but which rules governed it.

I6 — Outcome Determinism

Given identical **Payload Kind**, inputs, evidence references, policy bundles, and configuration state, governance outcomes **SHOULD be reproducible within declared bounds of nondeterminism**.

If nondeterminism is present, it **MUST be explicitly documented and traceable** through Decision Receipts.

Associated **Reason Codes MUST remain consistent** with the reproduced outcome.

Rationale:
Determinism (or declared nondeterminism) is required for debugging, auditing, and regulatory review.

I7 — Tamper Awareness

Decision Receipts **MUST include integrity mechanisms appropriate to the declared compliance profile**.

The Protocol defines the following integrity expectations:

- **RETNA-A:** basic integrity awareness (e.g., append-only logging recommended)
- **RETNA-B:** tamper-evident mechanisms **REQUIRED**
- **RETNA-C:** cryptographic attestation or strongly equivalent integrity guarantees **REQUIRED**

Any detected integrity failure **MUST be recorded and flagged**.

Rationale:
Receipts that can be altered without detection cannot serve as trust artifacts.

I8 — Separation of Duties

Decision Construction and Decision Governance **SHOULD be separable functions**, even if implemented within the same runtime or service.

Implementations **MUST NOT require governance logic to be embedded within model inference or reasoning processes**.

Rationale:
Separation of duties enables independent validation, reduces conflicts of interest, and supports multi-vendor interoperability.

5.1 Invariant Enforcement Scope

System Invariants apply to:

- all decisions within governed decision classes
- all compliance profiles (**RETNA-A, RETNA-B, RETNA-C**)
- all deployment environments (cloud, edge, embedded, hybrid)

Implementations **MAY apply stricter guarantees**, but **MUST NOT weaken or bypass these invariants**.

5.2 Invariant Violations

If any invariant is violated:

- the affected decision **MUST be treated as non-compliant**
- the system **MUST NOT claim R.E.T.N.A compliance** for that decision
- remediation **SHOULD be documented and auditable**

Transition Note (Non-Normative)

With the invariants defined, the following sections specify how systems fulfill these guarantees through roles, artifacts, and protocol flows.

System Invariants are presented before Actors and Roles intentionally, because the invariants define the guarantees that the subsequent roles and protocol mechanisms must satisfy.

---

6. Actors and Roles (Normative)

This section defines the actors recognized by the **R.E.T.N.A Protocol** and the responsibilities assigned to each.

Actors represent **functional roles**, not required deployment units. A single system component **MAY implement multiple roles**, provided that all applicable **System Invariants** are upheld.

For **governed decision classes**, any interaction that may result in execution **MUST pass through the Governance Boundary**.

6.1 Required Actors

The following actors are **REQUIRED** for any system claiming **R.E.T.N.A compliance**.

6.1.1 Producer

The **Producer** is any component that constructs and proposes a decision for potential execution.

Examples include:

- agent orchestrators
- planners
- workflow engines
- model routers
- task schedulers

Responsibilities

The Producer **MUST**:

- construct a **Decision Proposal** containing sufficient context, evidence references, and intent
- submit the Decision Proposal to the **Governance Boundary**
- await an explicit **Governance Outcome** prior to execution

Constraints

A Producer **MUST NOT**:

- execute a governed decision directly
- bypass or suppress governance evaluation
- forge or modify **Decision Receipts**

Invariant Mapping: I1, I2, I3, I8

6.1.2 Governor (R.E.T.N.A Implementation)

The **Governor** is the component that implements **R.E.T.N.A governance logic** and enforces the **Governance Boundary**.

The Governor **is the authoritative decision authority for governed decision classes.**

The Governor **MAY** be implemented as:

- a standalone service
- an embedded library
- a policy enforcement point within an execution gateway

Responsibilities

The Governor **MUST**:

- receive **Decision Proposals** from Producers
- evaluate proposals against policies, constraints, and evidence
- determine a deterministic **Governance Outcome**
- emit a **Decision Receipt** for every evaluated proposal
- communicate authorization (or denial) to the **Executor**

Constraints

The Governor **MUST**:

- be authoritative over execution authorization
- record all evaluated decisions, including denied or deferred proposals
- apply policies consistently with the declared **compliance profile**

Invariant Mapping: I1, I2, I5, I6, I7

6.1.3 Executor

The **Executor** is any component capable of applying a decision to the environment.

Examples include:

- payment processors
- IoT actuators
- database mutation services
- notification dispatchers
- access control systems

Responsibilities

The Executor **MUST**:

- verify governance authorization prior to execution
- validate the authenticity and integrity of the **Decision Receipt** where applicable
- execute only decisions explicitly authorized by the **Governor**
- report execution status where applicable

Constraints

An Executor **MUST NOT**:

- execute a governed decision without authorization
- treat missing authorization as recoverable
- fabricate or alter **Decision Receipts**

Lack of authorization **MUST be treated as a hard failure**.

Invariant Mapping: I1, I2, I7

6.1.4 Receipt Store

The **Receipt Store** is a durable storage system responsible for persisting **Decision Receipts**.

Examples include:

- databases
- append-only logs
- audit ledgers
- compliance data stores

Responsibilities

The Receipt Store **MUST**:

- persist all **Decision Receipts** emitted by the Governor
- preserve receipt integrity according to the declared **compliance profile**
- support retrieval and export for audit and investigation

Constraints

The Receipt Store **MUST NOT permit undetectable modification of stored receipts**.

Receipt storage **MUST provide tamper-aware durability consistent with the declared compliance profile.**

Receipt deletion or alteration **MUST be governed by policy**.

Invariant Mapping: I2, I7

6.2 Optional Actors

The following actors are **OPTIONAL but RECOMMENDED** for enterprise and regulated deployments.

6.2.1 Policy Authority

The **Policy Authority** defines, versions, and publishes **Policy Bundles** evaluated by the Governor.

Responsibilities

The Policy Authority **MUST**:

- publish identifiable, versioned policy bundles
- ensure policy immutability once published
- provide policy provenance and ownership metadata

Constraints

Policy bundles **SHOULD be independently auditable**.

Policy updates **SHOULD follow change-management practices**.

Invariant Mapping: I5, I6

6.2.2 Evidence Store

The **Evidence Store** holds raw or derived **Evidence Artifacts** referenced by **Decision Receipts**.

Examples include:

- document stores
- sensor data repositories
- feature stores
- snapshot archives

Responsibilities

The Evidence Store **MUST**:

- store evidence artifacts referenced by receipts
- provide integrity references (hashes or immutable identifiers)
- enforce access and retention policies

Constraints

Evidence Stores **SHOULD minimize exposure of sensitive data**.

Evidence retrieval **MUST be auditable where required**.

Invariant Mapping: I4, I7

6.2.3 Attestor

The **Attestor** provides cryptographic or strongly equivalent integrity guarantees for **Decision Receipts**.

Responsibilities

The Attestor **MUST**:

- sign or otherwise attest to receipt integrity
- support verification by third parties where required

Constraints

Attestation mechanisms **MUST align with the declared compliance profile**.

Key management **MUST follow established security best practices**.

Invariant Mapping: I7

6.3 Role Composition and Separation

A single component **MAY implement multiple roles** (for example, **Governor + Receipt Store**) provided that all applicable **System Invariants** are preserved.

Implementations **SHOULD maintain logical separation between**:

- **Decision Construction (Producer)**
- **Decision Governance (Governor)**
- **Decision Execution (Executor)**

Rationale

Logical separation:

- enables independent validation
- reduces conflicts of interest
- supports multi-vendor interoperability

6.4 Role Identification and Traceability

All actors referenced in a **Decision Receipt MUST be identifiable via stable identifiers** sufficient to support:

- correlation across receipts
- audit and investigation
- version and provenance tracking

Identifiers **MAY include**:

- service identifiers
- component names
- version identifiers
- deployment environment identifiers

---

7. Decision Classification (Normative)

This section defines the mandatory **classification schema** applied to all governed decisions under the **R.E.T.N.A Protocol**.

Decision Classification establishes the structural context in which governance evaluation occurs. Before policies are evaluated, **the Governor MUST determine the decision's Risk Class, Action Type, and Decision Mode.** These classification attributes define the severity of potential outcomes, the operational authority required for execution, and the escalation pathways available to the system.

Decision classification establishes:

- the **risk envelope** of a proposed action
- the **execution modality** permitted
- the **policy strictness** required at the Governance Boundary

All **R.E.T.N.A-compliant implementations MUST classify each governed decision prior to policy evaluation**.

Classification **MUST remain stable during policy evaluation** and **MUST be recorded in the Decision Receipt**.

7.1 Classification Requirements

For every governed decision, the **Governor MUST assign the following classification fields**:

1. **Risk Class** (required)
2. **Action Type** (required)
3. **Decision Mode** (required)

These fields are **normative inputs to policy evaluation**.

All classification fields **MUST be included in the Decision Receipt**.

7.2 Risk Class (Required)

The **Risk Class** expresses the **maximum plausible impact** of executing the proposed decision, assuming worst-case conditions.

Risk classification **MUST be conservative** and **MUST NOT be downgraded to avoid policy enforcement**.

Risk classification **MUST consider the worst-case consequences of successful execution**.

7.2.1 Baseline Risk Classes

A governed decision **MUST be assigned exactly one** of the following risk classes.

Risk Class	Identifier	Description
Informational	R0_INFO	No direct external effect; advisory or observational only
Low Impact	R1_LOW	Reversible, non-financial, minimal consequence
Financial	R2_FINANCIAL	Commits funds, inventory, or contractual resources
Physical	R3_PHYSICAL	Actuates physical devices or controls environments
Safety-Critical	R4_SAFETY	Potential harm to persons, property, or regulated domains

---

7.2.2 Risk Assignment Rules

- Risk Class **MUST reflect potential impact**, not intent.

- Risk Class **MUST be assigned prior to policy evaluation**.
- Risk escalation **MAY occur during evaluation** if additional evidence indicates higher risk.
- Risk de-escalation **MUST be justified and recorded** in the Decision Receipt.

Invariant Mapping: I1, I6

7.3 Action Type (Required)

The **Action Type** identifies the semantic category of the proposed operation.

Action Types enable:

- policy reuse across systems
- regulator interpretation
- cross-domain analytics

7.3.1 Baseline Action Types

The following Action Types are **RECOMMENDED baseline categories**.

- NOTIFY
- REORDER
- DISCARD
- UNLOCK
- LOCK
- SHUT_OFF
- START
- SCHEDULE
- UPDATE_RECORD
- DELETE_RECORD
- AUTHORIZE_ACCESS
- CALL_EMERGENCY

Implementations **MAY introduce domain-specific Action Types**, provided they:

- preserve semantic clarity
- do not overload baseline meanings
- are recorded verbatim in the Decision Receipt

7.4 Decision Mode (Required)

The **Decision Mode** specifies the **level of autonomy permitted for execution**.

Decision Mode is an explicit governance control and **MUST be honored by the Executor**.

7.4.1 Decision Modes

A governed decision **MUST be classified as exactly one** of the following modes.

Mode	Identifier	Description
Autonomous	AUTONOMOUS	System executes without human confirmation
Assisted	ASSISTED	Human confirmation required prior to execution
Manual	MANUAL	System may recommend but MUST NOT execute

7.4.2 Decision Mode Constraints

- Decision Mode **MUST be compatible with Risk Class**.
- Higher Risk Classes **SHOULD default to reduced autonomy**.
- Decision Mode changes **MUST be recorded as governance outcomes** (for example via **DEGRADE**).

Invariant Mapping: I1, I8

7.5 Classification Integrity and Disclosure

The following rules apply to all compliant implementations.

- Classification fields **MUST be included in every Decision Receipt**.
- Classification **MUST be machine-verifiable**.
- Classification changes across retries or escalations **MUST be traceable**.

- Classification logic **SHOULD be policy-driven or declarative**.

Classification **MUST NOT be**:

- inferred post-hoc
- overridden silently
- omitted for performance reasons

7.6 Relationship to Policy Evaluation

Decision Classification:

- **precedes policy evaluation**
- **informs constraint severity**
- **determines escalation thresholds**

Policies **MAY reference**:

- specific Risk Classes
- Action Types
- Decision Modes
- combinations thereof

This separation ensures that **classification remains stable while policy evolves independently**.

8. The Decision Receipt (Normative Core)

The **Decision Receipt** is the primary artifact defined by the **R.E.T.N.A Protocol**.

All systems claiming **R.E.T.N.A compliance MUST emit a Decision Receipt for every governed decision**.

The **Governor** is the authoritative issuer of Decision Receipts for governed decision classes.

The Decision Receipt provides:

- **verifiable provenance**
- **enforceable accountability**
- **durable auditability**

for decisions constructed by AI or agentic systems and applied to real-world state.

A Decision Receipt is the canonical record of:

- what decision was proposed
- which actors participated
- which evidence was used
- which policies and constraints were evaluated
- which governance outcome was produced
- whether execution occurred
- how integrity and retention were preserved

The Decision Receipt therefore serves as the Protocol's **portable unit of AI accountability**.

8.1 Decision Receipt Requirements

A Decision Receipt **MUST**:

1. be emitted for every governed decision, regardless of outcome
2. capture sufficient information to explain what was proposed, why it was evaluated, and how the outcome was determined
3. be structured for both machine verification and human inspection
4. be persistable and retrievable according to the declared compliance profile

Decision Receipts **MUST be generated prior to execution for ALLOW outcomes** and **in lieu of execution for DENY, DEFER, ESCALATE, or DEGRADE outcomes**.

Decision Receipts **MUST** contain sufficient information to support:

- audit and investigation
- policy verification
- evidence traceability
- interoperability across compliant systems

Decision Receipts **MUST NOT** depend on access to proprietary model internals in order to be interpreted.

Invariant Mapping: I2, I3, I5, I7

8.2 Receipt Lifecycle

Every Decision Receipt **MUST progress through the following logical lifecycle states**:

1. **Proposed** — the decision has been submitted for governance evaluation
2. **Evaluated** — policies and constraints have been applied
3. **Resolved** — a Governance Outcome has been determined
4. **Executed** *(optional)* — the action has been applied
5. **Finalized** — the receipt has been sealed for audit and retained according to policy

Receipts **MAY**:

- be updated with execution status, or
- be linked to a follow-on execution receipt

provided that **trace continuity is preserved**.

If receipt state changes occur after initial issuance, those changes **MUST** preserve a verifiable relationship to the original receipt through explicit linkage or chaining.

A lifecycle transition **MUST NOT** erase prior state.

8.3 Canonical Receipt Structure (v0.1)

The following fields constitute the **minimum canonical structure** of a Decision Receipt.

Serialization format **is not mandated**.

Implementations **MUST define deterministic canonicalization rules** sufficient to ensure:

- stable hashing
- reproducible verification
- consistent attestation or signature behavior
- interoperable audit export across systems

Implementations **MAY** add additional fields, provided such additions do not alter the semantics of required canonical fields.

8.3.1 Receipt Header

The **Receipt Header** defines the identity and protocol versioning metadata of the receipt.

Field	Requirement
receipt_id	REQUIRED — globally unique
version	REQUIRED — protocol version
created_at	REQUIRED — ISO 8601 timestamp
expires_at	OPTIONAL — for deferred decisions

Additional header metadata **MAY** be included, provided canonical verification remains stable.

8.3.2 Identity Block

The **Identity Block** identifies the primary protocol actors associated with the receipt.

Actor	Fields
Producer	id, type, version
Governor	id, type, version, profile
Executor	id, type, version *(OPTIONAL if not executed)*

All actor identifiers **MUST** be stable enough to support:

- cross-receipt correlation
- audit and investigation
- provenance tracking

Identity records **MAY** include deployment or environment identifiers where necessary for audit.

8.3.3 Decision Envelope

The **Decision Envelope** describes the governed decision itself.

Field	Requirement
decision_id	REQUIRED — stable across retries
risk_class	REQUIRED
action_type	REQUIRED
decision_mode	REQUIRED
payload_kind	REQUIRED where supported by the implementation
proposed_action	REQUIRED — structured summary
target	REQUIRED — affected entity

The Decision Envelope **MUST** describe the candidate action independently of any internal model reasoning strategy.

If a **Payload Kind** is assigned, it **MUST** reflect the semantic category of the governed request and **MUST** remain consistent with downstream policy evaluation.

8.3.3.1 Payload Kind Semantics

Payload Kind identifies the semantic category of the governed request for routing, privilege determination, and policy applicability.

Payload Kind supplements the core classification triad of Risk Class, Action Type, and Decision Mode. It does not replace those fields and **MUST NOT be interpreted as a substitute for risk classification.**

Where supported by an implementation, Payload Kind **MUST be recorded in the Decision Envelope** and **MUST remain stable throughout policy evaluation for the evaluated request.**

Payload Kind **MAY** be used to:

- resolve policy applicability
- determine whether an edge or actor is authorized to carry a given request category
- distinguish governance artifacts, execution requests, evidence references, and observational payloads

Where a request belongs to a privileged category under local policy, ambiguous or missing Payload Kind resolution **MUST NOT authorize privileged execution.**

8.3.4 State Context

The **State Context** records the before/after references and the contextual state relevant to governance evaluation.

Field	Requirement
state_snapshot_ref_before	OPTIONAL
state_snapshot_ref_after	OPTIONAL
state_delta_summary	REQUIRED
environment_context	REQUIRED *(redactable)*

The State Context **MUST** provide sufficient information to explain what changed, or what was expected to change, if the governed action were executed.

Where raw state data is sensitive, implementations **SHOULD** prefer references and bounded summaries over embedded disclosure.

8.3.5 Participants

All materially participating entities involved in **Decision Construction MUST** be declared.

The **Participants Block MUST** support declaration of at least:

- agents_involved
- models_involved
- tools_invoked
- external_services

Each participant entry **MUST** include:

- a stable identifier
- a declared role or purpose
- a version, where applicable

The set of participants **MAY** also be represented through a **Dispatch Trace**, provided the canonical receipt remains interpretable independently.

Invariant Mapping: I3

8.3.6 Evidence Block

Each Decision Receipt **MUST reference the evidence used during Decision Construction and governance evaluation**.

Each evidence_item **MUST** include:

Field	Requirement
evidence_id	REQUIRED
type	REQUIRED
source	REQUIRED
timestamp	REQUIRED
hash	REQUIRED for Profiles B/C
location_ref	REQUIRED *(non-public)*
summary	REQUIRED *(bounded)*

Evidence items **MUST** be sufficiently specific to support retrieval, integrity validation, and audit review under the applicable compliance profile.

Where possible, evidence references **SHOULD** be preferred over raw embedded data.

Invariant Mapping: I4, I7

8.3.7 Policy Evaluation Block

The **Policy Evaluation Block** records the governance rules applied to the decision.

The receipt **MUST disclose all Policy Bundles applied**.

Each policy_bundle **MUST** include:

Field	Requirement
bundle_id	REQUIRED
bundle_version	REQUIRED
authority	REQUIRED

Each applied constraint **MUST** include:

- constraint_id
- result (pass / fail / unknown)
- severity
- reason_code

Risk assessment fields **MAY** include:

- risk_score
- risk_factors

Where applicable, the Policy Evaluation Block **SHOULD** record **Edge Policy** results governing whether the source actor was authorized to produce or route the decision.

Policy evaluation records **MUST** remain structured and machine-readable.

Invariant Mapping: I5, I6

8.3.8 Outcome Block

The **Outcome Block** records the final result of governance evaluation.

Field	Requirement
governance_outcome	REQUIRED
outcome_reason_codes	REQUIRED
required_human_confirmation	REQUIRED
fallback_path	OPTIONAL

Field	Requirement
execution_status	REQUIRED *(executed / not_executed / pending)*
human_override	REQUIRED if applicable

The Governance Outcome **MUST** be one of the canonical protocol outcomes:

- ALLOW
- DENY
- ESCALATE
- DEFER
- DEGRADE

At least one **Reason Code MUST** accompany every outcome.

Where human review, override, or fallback behavior occurs, the receipt **MUST** record that fact explicitly.

8.3.9 Integrity and Audit Block

The **Integrity and Audit Block** records the integrity protections and retention controls associated with the receipt.

Field	Requirement
receipt_hash	REQUIRED
previous_receipt_hash	OPTIONAL
signature	REQUIRED for Profile C
retention_class	REQUIRED
access_class	REQUIRED

Implementations **MUST** ensure that integrity metadata is sufficient to detect unauthorized modification of a receipt.

Where receipt chaining is used, previous_receipt_hash **MAY** be used to preserve continuity across related governance events.

Invariant Mapping: I7

8.3.10 Canonical Receipt Serialization

Implementations **MUST** define deterministic canonicalization rules sufficient to ensure stable hashing, reproducible verification, and interoperable audit export.

For JSON serialization, a conforming canonicalization profile **MAY** use the following rules:

- UTF-8 encoding
- canonical field names as defined by this section
- lexicographic key ordering
- no insignificant whitespace

Where a receipt hash is recorded, the receipt hash **MUST be computed over the canonical representation of the receipt with the "integrity.receipt_hash" field blanked or excluded prior to hashing.**

Receipt serialization **MAY** vary by implementation, but validators and auditors MUST be able to determine which canonicalization profile was used.

If an implementation supports multiple canonicalization profiles, the applied profile **MUST be recorded in the receipt integrity metadata.**

8.3.11 Canonical Receipt Example (Informative)

The following example illustrates a canonical Decision Receipt structure aligned with this specification.


```
{
 "receipt_id": "retna_8c1d0f8f08f2408b85f0f3650eb4e4e1",
 "version": "0.1",
 "created_at": "2026-03-15T01:00:00Z",
 "expires_at": null,
 "identities": {
  "producer": {
   "id": "device_integration_agent",
   "type": "agent",
   "version": "2.x"
```

```
  },
  "governor": {
    "id": "retna_governor",
    "type": "governor",
    "version": "0.1",
    "profile": "RETNA-A"
  },
  "executor": {
    "id": "central_ai_agent",
    "type": "agent",
    "version": "2.x"
  }
},
"decision_envelope": {
  "decision_id": "dec_123",
  "risk_class": "R2_FINANCIAL",
  "action_type": "REORDER",
  "decision_mode": "ASSISTED",
  "payload_kind": "execution_request",
  "proposed_action": {
    "summary": "reorder milk"
  },
  "target": {
    "id": "inventory_management_agent",
    "type": "agent"
  }
},
"state_context": {
  "state_snapshot_ref_before": null,
  "state_snapshot_ref_after": null,
  "state_delta_summary": {
    "item": "milk",
    "level": "low"
  },
  "environment_context": {
    "estimated_cost_usd": 4.99
  }
},
"participants": {
  "agents_involved": [],
  "models_involved": [],
  "tools_invoked": [],
  "external_services": []
},
"evidence": [],
"policy_evaluation": {
```

```
  "policy_bundles": [
   {
    "bundle_id": "RETNA_BASE",
    "bundle_version": "0.1",
    "authority": "VIS"
   }
  ],
  "constraints": []
 },
 "outcome": {
  "governance_outcome": "ALLOW",
  "outcome_reason_codes": [
   "RC_POLICY_PASS"
  ],
  "required_human_confirmation": false,
  "fallback_path": null,
  "execution_status": "authorized",
  "human_override": null
 },
 "integrity": {
  "profile": "RETNA-A",
  "canonicalization": "json:sort_keys,separators=(',',':'),utf8",
  "previous_receipt_hash": null,
  "receipt_hash": "<sha256>"
 },
 "dispatch_trace": {
  "trace_id": "trace_123",
  "path": [
   "device_integration_agent",
   "retna_governor",
   "central_ai_agent"
  ]
 }
}
```


This example is illustrative. It does not mandate a transport envelope, storage layout, or implementation language.

8.4 Receipt Immutability and Tamper Awareness

Decision Receipts **MUST be treated as append-only artifacts**.

- Receipts **MUST be tamper-evident**.
- Any mutation **MUST result in a new receipt or a chained reference**.
- Silent modification **constitutes a protocol violation**.

Receipts **MUST NOT** be altered in place in a manner that invalidates prior auditability.

Where receipt amendment is necessary, the amended state **MUST** be represented through an additional receipt or verifiable chain reference.

8.5 Minimal Receipt (RETNA-A)

Implementations claiming **RETNA-A** compliance **MUST emit, at minimum,** the following fields:

- receipt_id
- version
- created_at
- producer.id
- governor.id
- decision_id
- risk_class
- action_type
- decision_mode
- proposed_action summary
- state_delta_summary
- agents_involved
- policy_bundle identifiers
- governance_outcome
- reason_codes

Where an implementation supports **Payload Kind**, that field **SHOULD** be included in the minimal receipt profile as well.

The minimal receipt profile exists to preserve baseline accountability while allowing low-friction adoption of the Protocol.

Transition Note (Non-Normative)

With the Decision Receipt defined, R.E.T.N.A establishes a portable, attestable unit of AI accountability.

Subsequent sections specify:

- reason code standardization
- policy interfaces
- evidence handling
- interoperability
- conformance testing

---

9. Canonical Reason Codes (Normative)

This section defines the **baseline set of canonical reason codes** used by the **R.E.T.N.A Protocol** to explain governance outcomes in a structured and interoperable manner.

Reason codes provide a standardized explanation layer between **policy evaluation results** and the **interpretation of governance outcomes** by humans, systems, and auditors.

All systems claiming **R.E.T.N.A compliance MUST emit canonical reason codes when producing Decision Receipts**.

Reason codes enable:

- deterministic explanation of governance outcomes
- machine-verifiable audit trails
- interoperable interpretation across independent systems

Implementations **MAY extend the canonical reason code set**, but **MUST NOT alter the meaning or semantics of baseline codes defined by this Protocol**.

Reason codes are **normative artifacts** and form part of the **Decision Receipt Outcome Block**.

9.1 Purpose of Reason Codes

Canonical reason codes serve four primary purposes:

1. **Consistent interpretation of governance outcomes across systems**
2. **Support for automated audit, analytics, and alerting pipelines**
3. **Regulatory review without disclosure of proprietary model reasoning**
4. **Stable signals for downstream workflows**, including escalation, retry, fallback, or remediation

Reason codes are **not narrative explanations**.

They are **machine-readable assertions** that identify the **policy, risk, evidence, or system condition that influenced the governance outcome**.

Reason codes therefore act as the **interpretation interface between policy evaluation and external observability**.

9.2 Outcome Reason Codes (Baseline Set)

The following baseline reason codes are defined by the Protocol.

All **R.E.T.N.A-compliant implementations MUST support these codes**.

Additional reason codes **MAY be emitted in combination with these canonical codes**, provided canonical codes remain present when applicable.

9.2.1 Policy Evaluation Codes

These codes describe the outcome of policy constraint evaluation.

Code	Meaning
RC_POLICY_PASS	All applicable constraints evaluated successfully
RC_POLICY_FAIL	One or more blocking constraints failed
RC_POLICY_PARTIAL	Non-blocking constraints failed or warnings were issued

Policy evaluation codes **SHOULD accompany every Decision Receipt** unless a higher-level failure prevented evaluation.

9.2.2 Evidence and Confidence Codes

These codes describe conditions related to the **availability, freshness, or integrity of evidence inputs**.

Code	Meaning
RC_INSUFFICIENT_EVIDENCE	Required evidence was missing or incomplete
RC_LOW_CONFIDENCE	Confidence threshold required for action was not met
RC_EVIDENCE_STALE	Evidence used during evaluation was outdated or expired
RC_EVIDENCE_INTEGRITY_FAIL	Evidence integrity verification failed

Evidence codes **SHOULD be emitted when evidence deficiencies materially influenced the governance outcome**.

9.2.3 Risk and Safety Codes

These codes describe governance decisions influenced by **risk evaluation or safety constraints**.

Code	Meaning
RC_RISK_THRESHOLD_EXCEEDED	Computed risk exceeded allowed policy threshold
RC_SAFETY_GUARDRAIL	Safety constraint or guardrail was triggered
RC_REGULATORY_RESTRICTION	Regulatory or compliance rule prevented execution

Risk and safety codes **MUST be emitted when a governance decision was influenced by safety, compliance, or risk thresholds**.

9.2.4 System and Dependency Codes

These codes describe system conditions or environmental dependencies that influenced the outcome.

Code	Meaning
RC_EXTERNAL_DEPENDENCY_FAILURE	Required external system was unavailable or failed
RC_RATE_LIMITED	Execution deferred due to rate limiting or throughput constraints

Code	Meaning
RC_SYSTEM_DEGRADED	System health condition triggered reduced autonomy

System reason codes **SHOULD be emitted when governance decisions are influenced by operational system state**.

9.2.5 Governance Control Codes

These codes describe governance decisions related to **human oversight, escalation, or fallback behavior**.

Code	Meaning
RC_REQUIRES_CONFIRMATION	Human confirmation required before execution
RC_ESCALATION_REQUIRED	Escalation to a higher authority required
RC_FALLBACK_APPLIED	Safe fallback path selected
RC_HUMAN_OVERRIDE	Human operator overrode the governance outcome

Governance control codes **MUST be emitted when governance outcomes involve escalation, confirmation, or override behavior**.

9.3 Constraint Severity Codes

Constraint severity defines how a policy constraint influences the final governance outcome.

Severity precedence MUST be applied when multiple constraints influence the same decision.

The following severity levels are normative.

Severity	Meaning
INFO	Informational only; no effect on governance outcome

Severity	Meaning
WARN	Outcome allowed but flagged
BLOCK	Outcome MUST be denied or escalated
DEFER	Outcome MUST be deferred pending evidence or time
DEGRADE	Outcome allowed only with reduced autonomy

Constraint severity **MUST be recorded for each evaluated constraint within the Decision Receipt Policy Evaluation Block**.

Severity classification enables systems and auditors to distinguish between:

- blocking policy failures
- warning-level policy signals
- advisory constraints

9.4 Reason Code Usage Rules

The following rules apply to all compliant implementations.

- **At least one outcome reason code MUST appear in every Decision Receipt.**
- Reason codes **MUST be deterministic given identical inputs, policy bundles, and configuration state.**
- Reason codes **MUST be sufficient to justify the governance outcome.**
- Reason codes **MUST NOT expose proprietary model internals or private reasoning artifacts.**

Reason codes **SHOULD be**:

- concise
- enumerable
- stable across protocol versions
- interpretable without system-specific context

Where multiple factors influenced the outcome, **multiple reason codes MAY be emitted**.

9.5 Vendor Extensions

Vendors **MAY define additional reason codes** provided that:

- canonical codes are always preserved
- extended codes **do not redefine baseline meanings**
- extended codes **are namespaced**

Recommended namespacing format:

VENDORNAME_REASON_CODE

Example:

ACME_MODEL_TIMEOUT
ACME_SENSOR_DRIFT

Extended reason codes **MUST be clearly distinguishable from canonical protocol codes in the Decision Receipt**.

9.6 Relationship to Audit and Compliance

Canonical reason codes enable:

- **cross-system audit correlation**
- **automated compliance validation**
- **regulator-friendly inspection**
- **post-incident root cause analysis**

Because reason codes are standardized, **Decision Receipts can be interpreted across vendors, deployments, and domains without prior knowledge of system internals**.

This standardization ensures that governance behavior remains **transparent, explainable, and verifiable** even in heterogeneous AI ecosystems.

Transition Note (Non-Normative)

With reason codes standardized, the Protocol now defines:

- what was proposed
- how it was classified
- which policies were applied
- why it was allowed, denied, escalated, deferred, or degraded

The next section specifies how **governed decisions flow through the system**, making governance behavior observable, testable, and enforceable across implementations.

---

10. Protocol Flows (Normative)

This section defines the mandatory **execution flows** for governed decisions under the **R.E.T.N.A Protocol**.

Protocol flows specify:

- how decisions traverse the **Governance Boundary**
- how governance outcomes are enforced
- how **Decision Receipts** are emitted and finalized
- how continuity is preserved across retries, escalations, or deferrals

All systems claiming **R.E.T.N.A compliance MUST implement the applicable protocol flows corresponding to each governance outcome**.

Execution flows **MUST ensure that no governed decision can reach execution without first producing a valid Decision Receipt and governance outcome**.

10.1 Standard Flow — Governed Execution (ALLOW)

This flow represents the canonical path for a governed decision that is **authorized and executed**.

Flow Steps

1. Decision Construction

A **Producer** constructs a **Decision Proposal** containing:

- the proposed action
- contextual state information
- evidence references
- intended target of the action

The proposal **MUST contain sufficient information for governance evaluation**.

---

2. Submission to Governance Boundary

The Producer **submits the Decision Proposal to the Governor**.

The Producer **MUST NOT execute the proposed action while governance evaluation is pending**.

Execution **MUST remain blocked until an explicit governance outcome is returned**.

---

3. Classification

The Governor **assigns the required Decision Classification fields**:

- Risk Class
- Action Type
- Decision Mode

These fields **MUST be recorded in the Decision Receipt**.

---

4. Policy Evaluation

The Governor evaluates all applicable **Policy Bundles and Constraints**, including:

- policy rules
- risk thresholds
- safety guardrails
- regulatory constraints

- dependency and system health signals

Constraint evaluation results **MUST be recorded within the Decision Receipt Policy Evaluation Block**.

---

5. Governance Outcome

If all blocking constraints pass, the Governor determines the outcome:

governance_outcome = ALLOW

Reason codes describing the evaluation results **MUST be included**.

---

6. Receipt Emission

A **Decision Receipt MUST be emitted prior to execution**.

The receipt **MUST contain**:

- classification fields
- policy evaluation results
- reason codes
- authorization context

---

7. Execution

The **Executor** receives authorization and **MUST verify the Decision Receipt** before performing the action.

The Executor **MUST NOT execute a governed decision without valid authorization**.

---

8. Finalization

Execution status **MUST be recorded** in the Decision Receipt.

The receipt **MUST then be finalized and persisted** in the **Receipt Store**.

Invariant Mapping: I1, I2, I6

10.2 Deny Flow (DENY)

This flow applies when execution **is not permitted**.

Flow Characteristics

- Execution **MUST NOT occur**.
- Decision Receipt emission **MUST still occur**.

Flow Steps

1. Decision Proposal submitted to Governor
2. Classification and policy evaluation performed
3. Governance outcome determined as:

governance_outcome = DENY

4. Decision Receipt emitted containing:

- failing constraints
- applicable reason codes
- execution_status = not_executed

5. Receipt persisted in the Receipt Store

Denied decisions **MUST remain observable through their Decision Receipts**.

Invariant Mapping: I1, I2, I5

10.3 Escalation Flow (ESCALATE)

This flow applies when governance requires **review by a higher authority**.

Escalation Targets MAY Include

- human reviewer
- supervisory agent
- safety or compliance system
- external authorization service

Flow Requirements

Execution **MUST NOT occur until escalation is resolved**.

A Decision Receipt **MUST record the escalation event**.

Required Receipt Fields

The receipt **MUST contain**:

governance_outcome = ESCALATE

And include:

- escalation target identifier
- reason for escalation
- additional evidence requested (if applicable)

Escalation resolution **MAY produce**:

- a new Decision Receipt referencing the original, or
- a resolved outcome appended through receipt chaining.

Invariant Mapping: I2, I8

10.4 Defer Flow (DEFER)

This flow applies when execution is delayed due to **incomplete or unstable conditions**.

Common Deferral Causes

- insufficient evidence
- stale system state
- external dependency outage
- timing or cooldown rules

Flow Requirements

The Decision Receipt **MUST specify a path to resolution**.

Execution **MUST NOT occur until governance evaluation is re-run**.

Required Receipt Fields

The receipt **MUST contain**:

governance_outcome = DEFER

And include:

- expires_at or recheck_at
- required evidence list
- deferral reason codes

A deferred decision **MUST be re-evaluated through the Governance Boundary prior to execution**.

10.5 Degrade Flow (DEGRADE)

This flow applies when **execution may proceed but autonomy is reduced**.

Degradation Examples

- requiring human confirmation
- selecting a safer fallback action
- reducing operational scope
- limiting resource usage

Flow Requirements

Degradation **MUST be explicitly recorded in the Decision Receipt**.

Reduced autonomy **MUST be enforced by the Executor**.

Required Receipt Fields

The receipt **MUST contain**:

governance_outcome = DEGRADE

And include:

- fallback path
- autonomy restrictions
- confirmation requirements

Execution **MUST follow the reduced-autonomy constraints defined by the receipt**.

Invariant Mapping: I1, I8

10.6 Retry and Continuity Rules

When a decision is retried:

- decision_id **MUST remain stable**
- new receipts **MUST reference prior receipts**
- classification changes **MUST be disclosed**
- governance outcomes **MUST be independently justified**

Retries **MUST NOT overwrite or invalidate prior Decision Receipts**.

Silent retries without receipt emission **constitute a protocol violation**.

10.7 Failure Handling

If the **Governor becomes unavailable**:

- execution **MUST fail closed for governed decisions**
- no implicit allow behavior **is permitted**

If **receipt persistence fails**:

- execution **MUST NOT occur**
- the failure **MUST be observable to operators or monitoring systems**

Systems **MUST NOT execute governed decisions without a persisted Decision Receipt**.

10.8 Flow Determinism and Observability

Protocol flows **MUST remain observable through Decision Receipts**.

For identical inputs, evidence references, and policy bundles:

- governance outcomes **SHOULD be reproducible within declared bounds of nondeterminism**

Flow paths **MUST be auditable post hoc** without requiring access to internal model state.

Decision Receipts therefore provide the **complete observable history of governance behavior**.

Transition Note (Non-Normative)

With protocol flows defined, the **R.E.T.N.A Protocol now specifies**:

- what must happen
- when it must happen
- what evidence must exist afterward

The following section defines **how policies interface with governance evaluation**, ensuring modular policy design and interoperable enforcement across independent implementations.

11. Policy Interface (Normative)

This section defines the required interface between **governance logic** and **policy logic** within the **R.E.T.N.A Protocol**.

The R.E.T.N.A Protocol **does not mandate a specific policy language, rule engine, or constraint syntax**. Instead, the Protocol defines a **stable, interoperable policy interface** that allows policies to be authored, evaluated, versioned, and audited independently of implementation details.

This separation is **intentional and foundational**.

It ensures that:

- governance behavior remains **auditable and deterministic**
- policy authorship remains **independent of execution infrastructure**
- policy systems may evolve **without requiring protocol changes**

The Policy Interface therefore acts as the **contract between policy evaluation and governance enforcement**.

11.1 Policy Interface Objectives

The Policy Interface **MUST enable**:

- policy portability across systems and vendors
- deterministic governance outcomes
- transparent auditability without exposing proprietary logic
- independent evolution of policy and execution systems

The Policy Interface **MUST NOT**:

- assume a specific programming language
- require a particular rule engine or policy DSL
- embed policy logic directly within execution components

Policy evaluation **MUST occur within the Governance Boundary** and **MUST remain separable from Decision Construction and Decision Execution**.

11.2 Policy Bundle Concept

Policies are grouped into **Policy Bundles**.

A **Policy Bundle** is a **versioned, immutable collection of constraints** evaluated together during governance evaluation.

Policy Bundles allow:

- modular policy deployment
- independent policy governance
- regulatory overlays
- version-controlled policy evolution

A governance evaluation **MUST reference the Policy Bundles applied**.

11.2.1 Policy Bundle Requirements

Each Policy Bundle **MUST include**:

Field	Requirement
bundle_id	Stable unique identifier
bundle_version	Immutable version identifier
authority	Publisher or owning entity
effective_scope	Domains, risk classes, or action types covered

Policy Bundles **MUST be immutable once published**.

Updates **MUST result in a new bundle version**.

Implementations **MAY maintain multiple active bundle versions**, provided that the applied version **is disclosed in the Decision Receipt**.

11.3 Constraints

A **Constraint** is the smallest enforceable policy unit.

Constraints represent **individual evaluable rules** within a Policy Bundle.

Constraints **MUST be**:

- deterministic
- side-effect free
- independently evaluable

Constraints **MUST NOT perform execution actions or modify system state**.

Constraints exist solely to **inform governance decisions**.

11.3.1 Constraint Interface

Each constraint evaluation **MUST return the following fields**:

Field	Description
result	pass | fail | unknown
severity	INFO | WARN | BLOCK | DEFER | DEGRADE
reason_code	canonical or extended reason code
explanation_ref	optional structured reference

explanation_ref **MUST NOT contain free-form narrative text**.
It **SHOULD reference structured artifacts such as documentation IDs, policy references, or audit entries**.

Constraints **MUST NOT**:

- directly trigger execution
- mutate system state
- override governance outcomes outside defined severity semantics

11.4 Policy Evaluation Contract

The **Governor MUST evaluate all applicable Policy Bundles** for a given decision.

Policy applicability **MUST be resolved before constraint evaluation**.

For each constraint evaluated, the Governor **MUST record**:

- constraint identifier
- evaluation result
- severity classification
- associated reason code

The Governor **MUST determine the governance outcome by applying the most restrictive applicable severity**.

Severity Precedence (Normative)

The following severity precedence **MUST be applied**:

Priority	Severity
1	BLOCK
2	DEFER
3	DEGRADE
4	WARN
5	INFO

This precedence ordering **is normative and MUST NOT be altered**.

11.5 Determinism and Reproducibility

For identical:

- Decision Classification
- evidence references
- Policy Bundles
- configuration state

policy evaluation results **SHOULD be reproducible**.

If nondeterminism exists, it **MUST be explicitly declared and traceable**.

This requirement supports:

- audit replay
- incident investigation
- regulatory verification
- governance debugging

11.6 Policy Disclosure in Decision Receipts

Decision Receipts **MUST disclose**:

- which Policy Bundles were applied
- which constraints were evaluated
- which constraints influenced the final governance outcome

Decision Receipts **MUST NOT embed**:

- raw policy logic
- proprietary scoring formulas
- internal decision trees
- implementation-specific rule syntax

This requirement ensures **governance transparency without intellectual property leakage**.

11.7 Policy Authority and Trust

Policy Bundles **MUST identify their issuing authority**.

Authorities **MAY include**:

- system operators
- enterprises
- regulators
- industry consortiums

Governors **MAY enforce trust rules**, including:

- accepting bundles only from approved authorities
- prioritizing regulatory policies over local policies
- applying domain-specific trust hierarchies

Trust evaluation itself **MAY be policy-driven**.

11.8 Policy Scope and Applicability

Policies **MAY apply conditionally** based on:

- Risk Class
- Action Type
- Decision Mode
- operational domain or environment
- actor identity

Policy scope resolution **MUST occur prior to constraint evaluation**.

Scope resolution decisions **MUST be auditable**.

11.9 Extensibility

Implementations **MAY extend the Policy Interface** to support advanced policy models, including:

- richer explanation references
- probabilistic constraints
- temporal policies
- cross-receipt correlation

Extensions **MUST NOT**:

- alter baseline constraint semantics
- bypass constraint severity precedence
- suppress required policy disclosures

All extensions **MUST remain compatible with the canonical Decision Receipt structure**.

Transition Note (Non-Normative)

With the Policy Interface defined, the **R.E.T.N.A Protocol establishes governance without lock-in**:

- policy authors remain independent
- execution systems remain replaceable

- governance enforcement remains deterministic

The next section specifies **how evidence is captured, referenced, and verified**, completing the core governance triangle:

classification → policy → evidence

---

12. Evidence Handling (Normative)

This section defines the requirements for **capturing, referencing, protecting, and verifying evidence** used during **Decision Construction** and evaluated at the **Governance Boundary**.

Within the R.E.T.N.A Protocol, **evidence is treated as a first-class governance input**, not merely as diagnostic logging.

Evidence referenced during governance evaluation **MUST therefore satisfy integrity, traceability, and auditability requirements defined by the applicable Compliance Profile**.

Evidence referenced by a Decision Receipt **constitutes the factual basis upon which the governance outcome was determined**.

12.1 Evidence Principles

All compliant implementations **MUST adhere to the following principles**.

1. Sufficiency

Evidence referenced by a Decision Receipt **MUST be sufficient to justify the resulting governance outcome**.

If evidence is incomplete or unverifiable, governance evaluation **MUST account for this condition**, potentially resulting in outcomes such as **DEFER, ESCALATE, or DENY**.

2. Minimization

Decision Receipts **SHOULD reference evidence rather than embed raw data whenever feasible**.

Embedding raw evidence **SHOULD be avoided when external references are sufficient for audit and verification purposes**.

3. Integrity

Evidence references **MUST be tamper-evident** at the integrity level defined by the applicable Compliance Profile.

Integrity guarantees **MAY include hashing, immutable storage, or cryptographic attestation mechanisms**.

4. Traceability

Evidence referenced during governance evaluation **MUST be linkable to the Decision Receipt that relied upon it**.

Evidence items **MUST include identifiers that allow auditors to trace the relationship between evidence and governance outcomes**.

12.2 Evidence Representation

Evidence **MAY be represented in one of two forms**.

1. Embedded Summary

A **bounded, non-sensitive representation of evidence** included directly in the Decision Receipt.

Embedded summaries **SHOULD contain only the minimum information necessary for audit interpretation**.

2. External Reference

A **pointer to evidence stored outside the Decision Receipt**.

External references **MAY include**:

- internal datastore identifiers
- URI-like references
- immutable object storage references
- ledger or archival storage references

External references **SHOULD be used for Compliance Profiles B and C**, where evidence integrity verification is required.

Decision Receipts **MUST explicitly indicate which representation form is used**.

12.3 Evidence Item Requirements

Each evidence item referenced by a Decision Receipt **MUST include the following fields**.

Field	Requirement
evidence_id	Stable unique identifier
type	Evidence category (see Section 12.6 Evidence Types (Baseline))
source	Originating system, sensor, or actor
timestamp	Time of capture or generation
location_ref	Internal reference or URI-like pointer
summary	Bounded, human-readable description

For **Compliance Profiles B and C**, each evidence item **MUST additionally include**:

Field	Requirement
integrity_ref	Hash or equivalent integrity verification reference

The integrity reference **MUST enable verification that the referenced evidence has not been altered since capture**.

12.4 Evidence Integrity and Verification

Evidence integrity guarantees vary by **Compliance Profile**.

12.4.1 Integrity Guarantees by Profile

Profile A — Basic Governance

Evidence storage **SHOULD support append-only logging**.

Integrity guarantees **MAY rely on system-level logging controls**.

Profile B — Enterprise Governance

Evidence storage **MUST be tamper-evident**.

Acceptable mechanisms **MAY include**:

- cryptographic hashes
- checksums
- immutable object storage
- write-once storage systems

Evidence integrity **MUST be verifiable after receipt emission**.

Profile C — Regulatory Governance

Evidence **MUST support cryptographic attestation or equivalent integrity anchoring**.

Acceptable mechanisms **MAY include**:

- digital signatures
- cryptographic proof chains
- distributed ledger anchoring
- trusted timestamping services

Integrity verification **MUST be independently reproducible by auditors**.

12.4.2 Verification Failures

If evidence integrity verification fails:

- the governance evaluation **MUST record the integrity failure**
- appropriate reason codes **MUST be emitted** (e.g., RC_EVIDENCE_INTEGRITY_FAIL)
- execution **MUST NOT proceed for governed decision classes**

Integrity verification failure **SHOULD be treated as a high-risk anomaly condition**.

12.5 Evidence Lifecycle and Retention

Evidence storage and access **MUST respect declared governance policies**, including:

- retention_class
- access_class
- applicable regulatory or contractual requirements

Evidence **MAY be handled independently of Decision Receipts**, provided that receipt integrity remains verifiable.

Permitted lifecycle operations **MAY include**:

- retention in external storage systems
- redaction after receipt finalization
- archival storage migration
- controlled destruction according to retention policies

Receipt references **MUST remain valid for the declared retention period**.

12.6 Evidence Types (Baseline)

The following evidence types define **baseline categories** recognized by the R.E.T.N.A Protocol.

Implementations **MAY extend this list**, provided baseline semantics remain unchanged.

Type	Identifier	Description
Sensor	SENSOR	Physical or digital sensor readings
User Input	USER_INPUT	Explicit user instructions or confirmations
State Snapshot	STATE_SNAPSHOT	Captured system or environment state
Retrieved Document	RETRIEVED_DOC	Retrieved reference material (e.g., RAG output)
Derived Signal	DERIVED_SIGNAL	Computed features, scores, or anomaly signals
External Event	EXTERNAL_EVENT	Webhooks, alerts, or third-party signals

Additional evidence types **MAY be introduced through extension mechanisms**.

12.7 Evidence Minimization and Privacy

To protect privacy and reduce exposure of sensitive data:

Decision Receipts **MUST NOT embed sensitive raw data when references are sufficient**.

Evidence summaries **MUST be bounded and redactable**.

Access to raw evidence **MUST be governed independently from receipt visibility**.

Implementations **SHOULD support**:

- selective disclosure mechanisms
- redaction policies
- differential access control based on role or authority

Privacy protections **MUST NOT compromise the integrity or traceability of governance decisions**.

12.8 Evidence and Reproducibility

Where feasible, evidence references **SHOULD enable**:

- re-evaluation of governance decisions
- audit replay
- post-incident forensic analysis

Evidence references **SHOULD remain accessible for the declared retention period**.

If full reproduction is not possible due to data volatility or external system constraints, the Decision Receipt **MUST disclose this limitation**.

This disclosure **MUST allow auditors to understand the evidentiary limitations of the decision**.

Transition Note (Non-Normative)

With evidence handling defined, the protocol now specifies:

- what governance decisions rely upon
- how that reliance is recorded
- how evidence integrity is preserved over time

The next section formalizes **trust, security, and privacy guarantees**, ensuring that the protocol remains **defensible in adversarial or regulated environments**.

---

13. Trust, Security, and Privacy (Normative)

This section defines the **minimum trust, security, and privacy guarantees** required of any system claiming **R.E.T.N.A Protocol compliance.**

The R.E.T.N.A Protocol treats **governance as a security-critical control plane**.
Failures of governance enforcement **MUST therefore be treated as safety-critical failures**, not merely operational defects.

Systems implementing the protocol **MUST ensure that governance decisions cannot be bypassed, altered, or executed without proper authorization and receipt traceability.**

13.1 Authentication and Authorization

13.1.1 Actor Authentication

Actors participating in the governance workflow **MUST be authenticated prior to performing protocol actions.**

Specifically:

- The Governor **MUST authenticate Producers** submitting Decision Proposals.
- The Executor **MUST authenticate the Governor's authorization** prior to executing a governed action.

Acceptable authentication mechanisms **MAY include:**

- mutual TLS (mTLS)
- signed identity tokens
- hardware-backed identities
- cryptographic API credentials
- equivalent strong authentication mechanisms

Decision Proposals submitted by **unauthenticated or unverifiable actors MUST be rejected.**

13.1.2 Authorization Scope

Authorization **MUST enforce strict role separation** within the governance workflow.

The following authorization model **MUST be enforced**:

Role	Authorization
Producer	**MAY** construct and submit Decision Proposals
Governor	**MAY** evaluate policies and issue governance outcomes

Role	Authorization
Executor	**MAY** execute actions only when explicitly authorized

Producers **MUST NOT execute governed actions.**

Executors **MUST NOT perform policy evaluation.**

Governors **MUST NOT execute actions directly unless explicitly operating in a combined role with enforced logical separation.**

Privilege escalation **MUST be prevented by design.**

Invariant Mapping: **I1, I8**

13.2 Least Privilege and Separation of Duties

Implementations **MUST adhere to the principle of least privilege.**

Specifically:

- Producers **MUST NOT possess execution credentials**
- Executors **MUST NOT possess policy evaluation logic**
- Receipt Stores **MUST NOT permit unauthorized modification**

Where system components are **co-located within the same service or runtime, implementations MUST preserve logical separation of duties.**

This separation **MUST remain enforceable through authentication, authorization, or runtime isolation mechanisms.**

13.3 Tamper Resistance and Integrity

13.3.1 Receipt Integrity

All **Decision Receipts MUST be tamper-evident.**

Integrity guarantees **vary by Compliance Profile.**

Profile A

Append-only receipt storage **SHOULD be used.**

Profile B

Tamper-evident storage **MUST be implemented**, including mechanisms such as:

- cryptographic hashes
- immutable storage
- write-once logging

Profile C

Cryptographic attestation or equivalent ledger anchoring **MUST be implemented.**

If receipt tampering is detected, the system **MUST**:

- record the integrity violation
- surface the violation to governance monitoring systems
- treat the event as a governance failure

13.3.2 Evidence Integrity

Evidence integrity requirements are defined in **Section 12 (Evidence Handling) and MUST be enforced consistently with the declared Compliance Profile.**

Integrity verification failures **MUST influence governance outcomes.**

For governed decision classes, integrity verification failures **MUST prevent execution.**

13.4 Privacy Controls

13.4.1 Data Minimization

Implementations **MUST minimize exposure of personal or sensitive data.**

Decision Receipts **SHOULD reference evidence rather than embed raw data when possible.**

Evidence summaries **MUST be bounded and redactable.**

Access to raw data **MUST be governed independently of receipt visibility.**

13.4.2 Access Classification

Decision Receipts **MUST include an access_class field** describing the confidentiality level of the receipt.

Example classifications **MAY include**:

- public
- internal
- restricted
- regulatory

Access enforcement mechanisms **MUST be auditable**.

13.4.3 Retention Classification

Decision Receipts **MUST include a retention_class field** specifying retention requirements.

The retention classification **MUST define**:

- minimum retention duration
- maximum retention duration
- archival or deletion rules

Retention policies **MUST comply with**:

- contractual obligations
- regulatory requirements
- applicable Policy Bundles

13.5 Human Override and Accountability

Human intervention **MAY override governance outcomes** where operational or regulatory policies permit.

When a human override occurs:

- the override **MUST be recorded**
- the identity or role of the overrider **MUST be disclosed**, subject to privacy rules
- the justification **MUST be captured via reason codes**

Human overrides **MUST NOT**:

- erase or delete prior Decision Receipts
- silently alter governance outcomes
- bypass audit traceability requirements

Invariant Mapping: **I2, I7**

13.6 Threat Model (Baseline)

The R.E.T.N.A Protocol is designed to mitigate the following baseline threats:

- unauthorized execution of actions
- policy bypass attempts
- evidence forgery
- untraceable decision execution
- post-hoc alteration of decision history
- silent escalation of autonomous authority

This threat model **defines the minimum security assumptions of the protocol**.

Threats outside this baseline **MAY be addressed through domain-specific extensions**.

13.7 Fail-Closed Behavior

For governed decision classes, the system **MUST fail closed** when any of the following conditions occur:

- Governor unavailability
- Decision Receipt persistence failure
- integrity verification failure
- authentication failure
- authorization failure

Under fail-closed conditions:

- governed actions **MUST NOT execute**
- the failure condition **MUST be observable**

Implicit "allow" behavior **is a protocol violation**.

Transition Note (Non-Normative)

With trust, security, and privacy defined, the **R.E.T.N.A Protocol establishes governance enforcement as a secure control plane.**

The protocol therefore guarantees that governed decisions remain:

- enforceable
- auditable
- privacy-aware
- resilient under adversarial conditions

The remaining sections address **interoperability, conformance testing, and protocol evolution**, completing the protocol's path toward **standards-grade specification.**

---

14. Interoperability and Extensibility (Normative)

This section defines how **R.E.T.N.A-compliant systems interoperate across vendors, deployments, and domains**, and how the Protocol may be extended **without fragmentation or vendor lock-in**.

The R.E.T.N.A Protocol is designed as **a governance protocol rather than a product architecture**. Interoperability is therefore **a primary requirement**, not an implementation convenience.

Any system claiming protocol compliance **MUST preserve interoperability at the level of canonical governance artifacts**, regardless of internal system design.

14.1 Interoperability Principles

All compliant implementations **MUST adhere to the following principles**.

Semantic Stability

The meaning of **canonical protocol fields MUST remain consistent across implementations**.

Implementations **MUST NOT reinterpret or redefine the semantics of baseline protocol fields**.

Artifact Portability

Decision Receipts **MUST be interpretable outside the originating system**.

External systems **MUST be able to evaluate the structure, integrity, and governance outcome of a receipt without requiring proprietary internal logic**.

Vendor Neutrality

Protocol compliance **MUST NOT depend on proprietary software components**.

Any conforming system **MUST be able to validate Decision Receipts produced by another conforming implementation**.

Forward Compatibility

Protocol extensions **MUST NOT break baseline conformance**.

Implementations **MUST tolerate unknown non-critical fields** without failing validation.

14.2 Canonical Artifact Interoperability

14.2.1 Decision Receipt as the Interoperable Unit

The **Decision Receipt** is the **atomic unit of interoperability** within the R.E.T.N.A Protocol.

A system **MUST be considered interoperable at the protocol level** if it:

1. emits Decision Receipts conforming to **Section 8**, and
2. preserves all **required semantics and integrity guarantees**.

Protocol interoperability **does not depend on**:

- internal architecture
- programming language
- runtime environment
- deployment model

14.2.2 Serialization and Canonicalization

The R.E.T.N.A Protocol **does not mandate a specific serialization format**.

However, implementations **MUST define and publish canonicalization rules** sufficient to ensure cross-system verification.

Canonicalization rules **MUST specify**:

- canonical field ordering
- normalization rules for values
- canonical hashing inputs

These rules **MUST enable**:

- consistent receipt hashing
- cross-system verification
- independent third-party audit validation

14.2.3 Receipt Validation and Canonical Verification (Normative)

A conforming implementation **MUST** support validation of Decision Receipts at the artifact layer.

Receipt validation **MUST** be capable of verifying at least the following:

- presence of all required canonical receipt fields
- presence of required substructures for identities, decision envelope, policy evaluation, outcome, integrity metadata, and dispatch trace
- canonical hash correctness where a receipt hash is present
- profile-specific integrity expectations applicable to the declared compliance profile

A validator **MUST** reject receipts that:

- omit required governance-critical fields
- omit required actor identity disclosure
- omit a governance outcome
- omit integrity metadata required by the declared profile
- contain an invalid canonical receipt hash

Validation behavior **MUST be based on observable receipt content and MUST NOT require access to proprietary model internals.**

14.3 Vendor Extensions

14.3.1 Permitted Extensions

Implementations **MAY extend the protocol** by introducing additional fields or categories, including:

- additional evidence types
- additional reason codes
- domain-specific constraints
- supplemental risk metrics
- auxiliary metadata fields

Extensions **MUST remain compatible with baseline protocol semantics**.

14.3.2 Extension Constraints

Extensions **MUST NOT**:

- alter the meaning of baseline protocol fields
- omit required canonical fields
- redefine baseline reason codes
- bypass governance outcomes
- interfere with Decision Receipt integrity verification

All extensions **MUST be**:

- namespaced
- explicitly declared
- safely ignorable by baseline validators

Unknown extension fields **MUST NOT invalidate otherwise compliant Decision Receipts**.

14.3.3 Payload Kind Registry

Payload Kinds constitute a controlled vocabulary governing semantic request categories used for routing, privilege determination, and policy applicability.

Implementations supporting Payload Kind **MUST** define a registry or equivalent controlled list of recognized values.

Payload Kind registries **MUST** preserve the following rules:

- baseline meanings **MUST** remain stable
- privileged categories **SHOULD** be explicitly declared
- unknown or ambiguous privileged categories **MUST** fail closed where required by policy
- implementation-specific extensions MUST be documented and **SHOULD** be namespaced when shared across system boundaries

Changes to baseline Payload Kind semantics constitute protocol-level changes and **MUST NOT be introduced silently.**

14.3.4 Registry-Governed Extension Points

The following protocol surfaces are registry-governed extension points:

- canonical reason codes
- payload kinds
- evidence type extensions
- action type extensions
- domain profile identifiers

For all such extension points:

- baseline protocol semantics **MUST NOT** be redefined
- required canonical fields **MUST NOT** be bypassed
- implementation-specific values **SHOULD** be documented in machine-readable form where practical
- unknown non-critical values **MAY** be ignored only where such behavior is safe
- unknown privileged categories **MUST fail closed where required by policy**

This section does not require a specific governance body for registry management in v0.1, but implementers **SHOULD** maintain transparent versioned records of registry changes.

14.4 Capability Advertisement

Governors **SHOULD advertise their protocol capabilities** in a machine-readable form.

Capability descriptions **MAY include**:

- supported compliance profile (A, B, C)
- supported policy bundle formats
- supported attestation mechanisms
- supported receipt export formats

Capability discovery **enables**:

- automated system integration
- toolchain compatibility
- cross-vendor orchestration

14.5 Cross-System Validation

Any third-party system **SHOULD be able to**:

1. ingest a Decision Receipt
2. verify structural protocol compliance
3. validate integrity claims
4. interpret the governance outcome

This validation **MUST NOT require access to proprietary internal decision logic**.

Cross-system validation is essential for:

- regulators
- auditors
- insurance providers
- incident response teams

14.6 Backward and Forward Compatibility

14.6.1 Backward Compatibility

Minor protocol versions **MUST**:

- preserve required canonical fields
- preserve semantic meaning of existing fields
- allow previously issued Decision Receipts to remain valid

Breaking semantic changes **MUST only occur in major protocol revisions**.

14.6.2 Forward Compatibility

Implementations **MUST ignore unknown fields** unless those fields are explicitly marked as **critical extensions**.

This requirement enables:

- gradual ecosystem adoption

- safe experimentation with extensions
- protocol evolution without disruption

14.7 Domain Profiles (Optional)

Industries **MAY define domain-specific profiles** layered on top of the R.E.T.N.A Protocol.

Example domain profiles include:

- healthcare governance profile
- financial services governance profile
- industrial automation governance profile
- smart infrastructure governance profile

Domain profiles **MUST**:

- inherit baseline protocol invariants
- declare any additional constraints
- remain interoperable at the Decision Receipt level

Domain profiles **MUST NOT alter baseline protocol semantics**.

Transition Note (Non-Normative)

With interoperability and extensibility defined, the R.E.T.N.A Protocol establishes a **stable governance core with controlled evolution**, a prerequisite for long-term industry adoption.

The next section defines **conformance and compliance verification**, completing the protocol's enforceability loop.

---

15. Conformance and Testability (Normative)

This section defines the requirements for **demonstrating conformance to the R.E.T.N.A Protocol** and establishes **testable criteria** for validation by:

- internal engineering teams

- third-party auditors
- regulatory authorities

A system **MUST NOT claim R.E.T.N.A Protocol compliance** unless it can demonstrably satisfy the requirements defined in this section.

Conformance verification **MUST be based on observable protocol behavior**, not on internal implementation details.

15.1 Conformance Claims

A system **MAY claim conformance to the R.E.T.N.A Protocol only if it satisfies all of the following conditions**:

1. It **implements all Normative (MUST/SHALL) requirements** applicable to its declared Compliance Profile.
2. It **emits Decision Receipts conforming to Section 8**.
3. It **upholds all System Invariants (I1–I8)** defined by the protocol.
4. It **passes the conformance tests defined in this section**.

Conformance declarations **MUST include**:

Field	Requirement
protocol_version	Implemented protocol version (e.g., RETNA v0.1)
compliance_profile	One of: RETNA-A, RETNA-B, or RETNA-C
governed_scope	Declared scope of governed decision classes

Systems **MUST NOT advertise protocol compliance without declaring these attributes**.

15.2 Mandatory Conformance Tests

The following conformance tests **MUST be supported by all compliant implementations**.

These tests validate that the **governance control plane cannot be bypassed and remains observable**.

15.2.1 Governance Boundary Integrity Test

Objective

Verify that governed decisions cannot execute without governance evaluation.

Test Procedure

Attempt to execute a governed action **without submitting a Decision Proposal to the Governor**.

Expected Result

- Execution **MUST fail**.
- No system side effects **MUST occur**.
- The failure **MUST be observable**.

Invariant Mapping: **I1**

15.2.2 Receipt Emission Test

Objective

Verify that all governed decisions produce a Decision Receipt.

Test Procedure

Submit governed decisions that result in each possible governance outcome:

- ALLOW
- DENY
- ESCALATE
- DEFER
- DEGRADE

Expected Result

- Exactly **one Decision Receipt MUST be emitted per governance evaluation**.
- Receipts **MUST persist regardless of outcome**.

Invariant Mapping: **I2**

15.2.3 Receipt Completeness Test

Objective

Verify that Decision Receipts include all required canonical fields.

Test Procedure

Inspect emitted receipts for required fields defined in **Section 8**.

Expected Result

- All required canonical fields **MUST be present**.
- Silent omission of required fields **MUST NOT occur**.

15.2.4 Policy Trigger Test

Objective

Verify correct constraint enforcement during governance evaluation.

Test Procedure

Introduce a policy constraint designed to fail.

Expected Result

- The governance outcome **MUST reflect the constraint severity**.
- Appropriate **reason codes MUST be emitted**.

Invariant Mapping: **I5**

15.2.5 Deterministic Outcome Test

Objective

Verify governance reproducibility.

Test Procedure

Submit identical Decision Proposals containing:

- identical evidence references
- identical policy bundles
- identical configuration state

Expected Result

- Governance outcomes **SHOULD be identical**.
- Any nondeterminism **MUST be explicitly declared**.

Invariant Mapping: **I6**

15.2.6 Receipt Validator Conformance Test

Objective

Verify that a conforming validator can correctly accept valid receipts and reject materially invalid receipts.

Test Procedure

Provide the validator with:

1. a structurally complete receipt with a valid canonical hash
2. a receipt missing one or more required canonical fields
3. a receipt with an invalid canonical hash
4. a receipt missing required profile-specific integrity elements for the declared profile

Expected Result

The validator **MUST**:

- accept the valid receipt
- reject the structurally incomplete receipt
- reject the receipt with an invalid canonical hash
- reject the receipt that fails profile-specific integrity requirements

Validation failure **MUST be observable and machine-readable.**

15.3 Profile-Specific Tests

Additional conformance tests apply depending on the declared **Compliance Profile**.

15.3.1 Profile B (RETNA-B) Tests

Implementations claiming **RETNA-B** compliance **MUST additionally pass the following tests**.

Evidence Integrity Test

Test

Introduce an evidence hash mismatch.

Expected Result

- The mismatch **MUST be detected**.
- The failure **MUST be recorded in the Decision Receipt**.

Tamper-Evident Storage Test

Test

Attempt unauthorized modification of stored Decision Receipts.

Expected Result

- The modification **MUST be detectable**.

15.3.2 Profile C (RETNA-C) Tests

Implementations claiming **RETNA-C** compliance **MUST additionally pass the following tests**.

Attestation Verification Test

Test

Verify receipt signatures or ledger anchors.

Expected Result

- Attestation verification **MUST succeed for valid receipts**.

Human Override Recording Test

Test

Perform a governance override via authorized human intervention.

Expected Result

- The override **MUST produce explicit receipt entries**.

Third-Party Verifiability Test

Test

Provide receipts to an external validation system.

Expected Result

- The external validator **MUST be able to verify receipt integrity and governance outcomes**.

15.4 Negative Testing Requirements

Conformance testing **MUST include negative scenarios**, including:

- missing evidence
- malformed receipts
- unauthorized Producers
- unavailable Governor
- Decision Receipt persistence failure

For all such conditions, the system **MUST fail closed**.

Governed actions **MUST NOT execute under these failure conditions**.

15.5 Audit Replay Capability

Implementations **SHOULD support audit replay capabilities**, including:

- replay of Decision Receipts
- reconstruction of decision context (within privacy limits)
- export of receipts for regulatory or incident review

Replay capability **MAY be partial** where evidence volatility prevents full reconstruction, provided the limitation **is disclosed in the Decision Receipt**.

15.6 Independent Verification

A compliant implementation **SHOULD be independently verifiable** by:

- internal audit teams
- third-party assessors
- regulators

Verification **MUST be possible without access to**:

- proprietary model internals
- source code
- private policy logic

Decision Receipts **MUST provide sufficient information to enable verification of governance outcomes**.

Transition Note (Non-Normative)

With conformance and testability defined, the R.E.T.N.A Protocol now specifies **how governance trust is proven rather than merely asserted**.

The remaining sections address:

- implementation guidance
- protocol versioning and evolution
- ecosystem adoption strategies

These complete the protocol's transition from **conceptual governance framework to deployable infrastructure specification**.

---

16. Reference Implementation Guidance (Non-Normative)

This section provides **implementation guidance** for adopting the **R.E.T.N.A Protocol** in real-world systems.

This section is **non-normative** and **does not impose additional requirements** beyond those defined in earlier sections of this specification.

The purpose of this guidance is to:

- accelerate adoption
- reduce the risk of incorrect implementations
- demonstrate practical deployment patterns
- illustrate feasibility without introducing vendor lock-in

16.1 Architectural Placement

R.E.T.N.A is designed to operate at the **Governance Boundary**, positioned between **Decision Construction** and **Decision Execution**.

Several architectural deployment patterns are commonly used.

Gateway Pattern

The **Governor operates as an authorization gateway** positioned in front of execution systems.

Execution requests pass through the Governor before any action is performed.

Library Pattern

The Governor is implemented as a **governance library embedded inside an application**, invoked synchronously before execution.

This approach may be suitable for tightly integrated systems.

Service Pattern

The Governor operates as a **standalone governance service**, accessed via an API by Producers and Executors.

This pattern supports:

- distributed systems
- multi-agent architectures
- cross-service governance enforcement

All deployment patterns are acceptable provided that **System Invariants (I1–I8) remain preserved**.

16.2 Minimal Reference Components

A minimal reference implementation typically contains the following components.

1. Governance Interface

Accepts **Decision Proposals** and returns governance outcomes.

Typical responsibilities include:

- receiving Decision Proposals
- evaluating proposals against policies
- emitting Decision Receipts

---

2. Policy Evaluation Module

Responsible for:

- loading Policy Bundles
- evaluating Constraints
- producing reason codes
- determining constraint severity results

3. Receipt Store

Responsible for:

- persisting Decision Receipts
- enforcing immutability or tamper-evidence
- enabling audit retrieval

---

4. Evidence Reference Layer

Responsible for:

- generating evidence identifiers
- storing or referencing raw artifacts
- producing integrity hashes where required

---

5. Optional Attestation Module

Used primarily for **RETNA-C deployments**.

Responsibilities may include:

- signing Decision Receipts
- anchoring receipts to immutable ledgers
- producing cryptographic verification artifacts

16.3 Decision Proposal Interface (Illustrative)

A typical **Decision Proposal** may include:

- proposed action summary
- target entity or resource
- evidence references
- relevant environment context
- requesting Producer identity

Implementations typically ensure that:

- proposals are structured
- proposals are authenticated
- proposals are immutable once submitted

16.4 Receipt Emission Strategy

Decision Receipts are generally emitted **prior to execution** for governed decisions.

Typical emission patterns include:

Outcome	Emission Timing
ALLOW	emitted prior to execution
DENY	emitted immediately
DEGRADE	emitted prior to degraded execution

Outcome	Emission Timing
ESCALATE	emitted upon escalation
DEFER	emitted upon deferral

Execution systems typically **delay action until receipt emission succeeds**, ensuring governance traceability.

16.5 Receipt Storage Patterns

Common storage approaches for Decision Receipts include:

- append-only databases
- write-once object stores
- ledger-backed systems
- SIEM-integrated audit stores

Operational implementations often support:

- receipt retrieval by identifier
- filtering by risk class or outcome
- export for audit or regulatory review

16.6 Validation and Tooling

Implementers are encouraged to provide tooling such as:

- receipt schema validators
- integrity verification utilities
- receipt viewers or dashboards
- automated conformance test harnesses

These tools ideally operate **independently of proprietary execution logic**, allowing external validation.

16.6.1 Reference Validator Tooling

Reference implementations may provide tooling such as:

- receipt schema validators
- canonical hash verification utilities
- profile-aware receipt validators
- machine-readable registry exports for payload kinds and reason codes

Such tooling is particularly useful for CI, audit replay workflows, cross-system verification, and independent conformance assessment.

Where provided, reference tools **SHOULD remain protocol-first** and **SHOULD avoid coupling validation semantics to proprietary execution logic.**

16.7 HomeSphere AI as a Reference Implementation (Informative)

HomeSphere AI provides an example implementation demonstrating the R.E.T.N.A Protocol across:

- multi-agent orchestration
- physical device actuation
- real-time sensor evidence ingestion
- policy-driven governance evaluation
- auditable Decision Receipt generation

HomeSphere illustrates **one possible deployment architecture**.

R.E.T.N.A remains **independent of HomeSphere** and can be implemented by any system that satisfies the protocol requirements.

16.8 Common Implementation Pitfalls

Implementers commonly encounter several avoidable mistakes.

Examples include:

- treating Decision Receipts as operational logs rather than governance artifacts
- emitting receipts after execution instead of before execution

- embedding sensitive data unnecessarily within receipts
- silently retrying decisions without emitting new receipts
- coupling policy logic directly to execution logic

Avoiding these pitfalls improves **auditability, interoperability, and protocol compliance**.

---

17. Licensing and Implementation Rights (Normative)

The R.E.T.N.A Protocol Specification defines a proprietary decision governance framework developed and owned by Value Intelligence Solutions Inc.

This document is made available for informational, evaluative, and research purposes. Access to this specification does not grant any rights to implement, reproduce, modify, distribute, or commercialize the protocol or any derivative system without explicit authorization.

Implementations of the R.E.T.N.A Protocol in production, commercial, or operational environments **MAY** require a valid license issued by Value Intelligence Solutions Inc.

All conformant implementations **MUST preserve the core invariants, governance boundary enforcement, decision receipt requirements, and policy evaluation mechanisms defined in this specification.**

The following restrictions apply:

- Unauthorized implementation of the R.E.T.N.A Protocol for commercial or operational use is prohibited
- Creation of derivative governance frameworks that replicate the core architectural model, decision receipt structure, or policy enforcement mechanisms may be subject to enforcement
- Redistribution of this specification without proper attribution is not permitted

Permitted uses include:

- Internal evaluation and research
- Academic study and analysis
- Non-commercial prototyping for validation purposes

Organizations seeking to implement the R.E.T.N.A Protocol in production systems **SHOULD obtain a formal license or partnership agreement with Value Intelligence Solutions Inc.**

Value Intelligence Solutions Inc. reserves the right to define licensing tiers, compliance requirements, certification programs, and conformance validation processes for implementations of the protocol.

Nothing in this specification shall be interpreted as granting implicit rights to deploy, commercialize, or operationalize the R.E.T.N.A Protocol without authorization.

All trademarks, system designs, architectural patterns, and governance mechanisms described herein remain the exclusive intellectual property of Value Intelligence Solutions Inc.

---

18. Versioning and Evolution (Normative)

This section defines the **versioning model, compatibility rules, and evolution guarantees** of the **R.E.T.N.A Protocol**.

The protocol is intended to function as a **long-lived governance standard**.
Protocol evolution **MUST therefore remain predictable, transparent, and safe** for adopters operating in **regulated, safety-critical, or mission-critical environments**.

Protocol changes **MUST preserve the auditability and interpretability of previously issued Decision Receipts**.

18.1 Version Identifier Format

R.E.T.N.A protocol versions **MUST follow the format**:

major.minor

Example versions include:

0.1
1.0
1.1

Each **Decision Receipt MUST include the protocol version** under which the receipt was emitted.

This version identifier **enables validators to interpret receipts according to the appropriate protocol semantics**.

18.2 Minor Version Rules

Minor version increments (for example 0.1 → 0.2 or 1.0 → 1.1) represent **backward-compatible protocol evolution**.

Minor versions **MAY introduce**:

- additional optional fields
- clarifications of semantic definitions
- new canonical reason codes
- new evidence or action type categories
- expanded non-normative guidance

Minor versions **MUST NOT**:

- remove required canonical fields
- alter the meaning of existing fields
- break Decision Receipt integrity verification
- invalidate previously issued Decision Receipts

Receipts emitted under earlier minor versions **MUST remain valid and interpretable**.

18.3 Major Version Rules

Major version increments (for example 0.x → 1.0 or 1.x → 2.0) represent **significant protocol evolution**.

Major versions **MAY introduce**:

- new required fields
- removal or deprecation of legacy constructs
- new compliance profiles
- refinements to protocol invariants with explicit migration paths

Major versions **MUST**:

- publish explicit migration guidance
- preserve interpretability of previously issued Decision Receipts
- avoid silent semantic changes

Even when implementations evolve, **receipt verification and audit interpretation MUST remain possible across protocol versions**.

18.4 Deprecation Policy

When protocol elements are deprecated, including:

- fields
- reason codes
- protocol behaviors

the following rules apply:

- deprecation **MUST be explicitly declared** in the specification
- deprecated elements **MUST remain accepted for a defined compatibility period**
- Decision Receipts **SHOULD indicate deprecated usage where applicable**

Silent deprecation **MUST NOT occur**.

18.5 Receipt Compatibility Across Versions

Validators **MUST support cross-version receipt validation**.

Specifically, validators **MUST**:

- accept Decision Receipts emitted under earlier protocol versions
- ignore unknown non-critical fields
- preserve receipt hash verification using the canonicalization rules declared by the emitting version

Decision Receipts **MUST NOT require re-issuance solely due to protocol upgrades**.

Previously issued receipts **remain valid governance artifacts**.

18.6 Governance of the Protocol Itself

Evolution of the R.E.T.N.A Protocol **SHOULD occur through a transparent governance process**.

Such a process typically includes:

- public change proposals
- published rationale for protocol modifications
- versioned specification releases

Protocol stewardship **MAY be administered by**:

- an independent foundation
- an industry consortium
- a neutral steward entity with publicly declared governance rules

Regardless of the governing body, protocol evolution **SHOULD prioritize interoperability, stability, and audit continuity**.

18.7 Stability Guarantees

The R.E.T.N.A Protocol provides the following long-term stability guarantees.

1. Decision Receipts **remain auditable indefinitely**.
2. Governance outcomes **remain interpretable across protocol versions**.
3. Policy logic **remains decoupled from protocol evolution**.
4. Conformance claims **remain verifiable over time**.

Protocol changes **MUST preserve these guarantees**.

Transition Note (Non-Normative)

With versioning and evolution defined, the R.E.T.N.A Protocol ensures that:

- early adopters are protected from breaking changes
- regulated deployments remain valid across protocol revisions
- the protocol can evolve without fragmentation

The final section addresses **adoption strategy and ecosystem positioning**, completing the **R.E.T.N.A v0.1 specification**.

---

19. Adoption Path (Non-Normative, Strategic)

This section outlines a **practical staged adoption strategy** for the **R.E.T.N.A Protocol**, intended to accelerate industry uptake while preserving interoperability, regulatory trust, and long-term standardization potential.

The adoption model is designed to:

- minimize operational disruption
- reduce integration friction
- allow organizations to realize immediate value
- support gradual progression toward full governance enforcement

R.E.T.N.A can be adopted incrementally without requiring full system refactoring.

19.1 Adoption Philosophy

R.E.T.N.A adoption is guided by three core principles.

Receipts First

Adoption typically begins by **emitting Decision Receipts**, even before governance enforcement is enabled.

This allows organizations to establish **decision traceability and audit visibility** immediately.

Governance Gradually Enforced

Governance controls may initially operate in **observational mode**, with enforcement enabled progressively as confidence and operational maturity increase.

Protocol Before Platform

The protocol is designed to remain **valuable independently of any particular product or vendor implementation**.

Organizations should be able to implement the protocol using their existing systems and architectures.

19.2 Stage 0 — Observability-Only Adoption

Objective

Establish visibility into autonomous or semi-autonomous decisions without altering execution behavior.

Typical Characteristics

- Decision Receipts emitted alongside normal execution
- Governance outcomes recorded but not enforced
- Receipts analyzed for operational insights

Benefits

- no operational risk
- immediate decision audit trail
- organizational familiarity with governance concepts

19.3 Stage 1 — Governance Boundary Introduction

Objective

Introduce the **Governance Boundary** for selected decision classes.

Typical Characteristics

- decision risk classification enabled
- policy evaluation activated
- governance outcomes recorded but not enforced

Benefits

- visibility into policy violations
- improved operational awareness
- early regulatory readiness

19.4 Stage 2 — Selective Enforcement

Objective

Enable governance enforcement for higher-risk decisions.

Typical Characteristics

- blocking enabled for higher-risk decision classes (e.g., R3 / R4)
- escalation and deferral workflows active
- human confirmation required where appropriate

Benefits

- meaningful risk reduction
- improved compliance posture
- clear accountability boundaries

19.5 Stage 3 — Full Compliance Profiles

Objective

Achieve **RETNA-B or RETNA-C compliance**.

Typical Characteristics

- evidence integrity verification
- tamper-evident receipt storage
- cryptographic attestation or ledger anchoring
- third-party verification capability

Benefits

- regulator-grade governance
- audit defensibility
- improved insurance and liability posture

19.6 Tooling and Ecosystem Enablement

Adoption is typically accelerated when an ecosystem provides supporting tooling such as:

- Decision Receipt validators
- receipt visualization tools
- audit export utilities
- conformance test harnesses
- policy authoring and validation tools

Such tooling ideally remains **protocol-first and vendor-neutral**.

19.7 Industry Standardization Path

A potential long-term trajectory for protocol maturation may include:

1. public publication of the protocol specification
2. independent reference implementations
3. formation of a neutral stewardship body or working group
4. engagement with formal standards organizations
5. alignment with regulatory frameworks and guidance

The protocol has been designed to support such standardization efforts **without requiring architectural redesign**.

19.8 Strategic Positioning

The R.E.T.N.A Protocol positions **Decision Governance** as a foundational layer of the emerging AI infrastructure stack.

Within this stack, R.E.T.N.A can be understood as analogous to:

- **TLS** for secure transport
- **OAuth** for identity delegation
- **OpenTelemetry** for observability

In this context, **Decision Receipts function as a portable governance artifact**, enabling trust, accountability, and verification across organizational boundaries.

19.9 HomeSphere AI as a Catalyst (Informative)

HomeSphere AI provides a working example demonstrating how the R.E.T.N.A Protocol can be applied in a real-world system involving:

- multi-agent orchestration
- physical device actuation
- sensor-derived evidence inputs
- policy-driven governance decisions
- auditable Decision Receipt generation

HomeSphere serves as an **illustrative reference implementation**, not as a required dependency.

R.E.T.N.A remains **fully implementable by any system satisfying the protocol requirements**.

Closing Statement (Non-Normative)

R.E.T.N.A Protocol v0.1 introduces **Decision Governance as a first-class, enforceable, and auditable capability** for AI systems operating in real-world environments.

By centering governance on **portable Decision Receipts**, the protocol enables:

- trust without vendor lock-in
- accountability without pervasive surveillance
- innovation without compromising safety

The protocol provides a foundation upon which organizations, regulators, and industry partners can build **reliable and trustworthy autonomous systems**.

Appendix A — Glossary

Access Classification

A policy-defined categorization determining which actors or systems may access specific governance artifacts, evidence references, or Decision Receipts.

Action Type

A classification describing the type of action proposed by a decision, such as purchase, notification, device actuation, or modification of system access controls.

Actor

An entity participating in a governed decision process, including systems, services, agents, devices, or human participants.

Actor Authentication

The process of verifying the identity of an actor participating in a governed decision process.

Anomaly Signal

A derived signal indicating unusual, abnormal, or potentially unsafe conditions that may influence policy evaluation or risk classification.

Attestation

Optional cryptographic proof verifying the authenticity and integrity of a Decision Receipt or associated governance artifact.

Authorization Boundary

The enforcement interface at which a proposed decision must obtain authorization before execution.
In this Protocol, the Authorization Boundary is implemented as the Governance Boundary.

Authorization Scope

The defined set of actions, resources, or decision categories an authenticated actor is permitted to initiate or execute.

Backward Compatibility

The ability of newer protocol implementations to interpret and process artifacts created by earlier protocol versions.

Canonical Artifact

A governance artifact whose structure and semantics are defined by the Protocol and are intended to be interoperable across implementations.

Canonical Reason Code

A standardized machine-readable code representing the outcome or justification of a governance decision.

Canonical Serialization

The standardized representation of protocol artifacts ensuring consistent interpretation across systems.

Canonicalization Rules

The set of rules defining how protocol artifacts must be structured and serialized to ensure interoperability.

Capability Advertisement

A mechanism by which a system declares supported protocol features, compliance profiles, or extension capabilities.

Compliance Profile

A predefined configuration of protocol requirements specifying governance enforcement levels, evidence integrity guarantees, and audit capabilities.

Conformance Claim

A declaration that a system implementation satisfies the requirements of a specified protocol version and compliance profile.

Conformance Test

A test verifying that an implementation behaves according to protocol requirements.

Constraint

An enforceable rule evaluated during policy evaluation that restricts or conditions decision execution.

Constraint Evaluation

The process of evaluating individual constraints within a policy bundle.

Constraint Severity

A classification indicating the enforcement priority of a constraint.

Canonical severity levels:

BLOCK
DEFER
DEGRADE
WARN
INFO

Context Capture

The process of collecting relevant state, signals, and environmental information required for decision evaluation.

Decision Class

A classification grouping decisions according to operational domain or governance requirements.

Decision Construction

The process of forming a candidate action or plan using inputs, signals, models, or rules.

Decision Execution

The act of applying an authorized decision to the environment.

Decision Governance

The protocol-governed process by which candidate decisions are evaluated, authorized, and executed according to defined policies.

Decision Mode

A configuration defining the operational posture of a system, such as fully autonomous operation, human-in-the-loop governance, or advisory mode.

Decision Receipt

A structured artifact containing provenance information, evidence references, policy evaluations, authorization outcomes, and associated reason codes for a governed decision.

Deterministic Outcome Test

A conformance test verifying that identical inputs and policies always produce the same governance outcome.

Dispatch Trace

A graph-like representation of the actors and processes involved in constructing and authorizing a governed decision.

Domain Profile

A domain-specific extension of the protocol defining additional governance rules or compliance requirements.

Edge Policy

A governance rule specifying which actors may issue or execute particular categories of decisions.

Embedded Evidence Summary

A compact representation of evidence included directly within a Decision Receipt.

Evidence

Inputs used during decision construction or policy evaluation.

Evidence may include sensor signals, user commands, retrieved documents, state snapshots, or derived signals.

Evidence Identifier

A unique identifier assigned to an evidence artifact.

Evidence Integrity Reference

A reference or verification mechanism used to confirm the integrity of stored evidence.

Evidence Integrity Verification

The process of confirming that referenced evidence has not been altered or corrupted.

Evidence Item

A single piece of evidence used in decision construction or policy evaluation.

Evidence Lifecycle

The full lifecycle of evidence artifacts including creation, reference, verification, retention, and eventual expiration.

Evidence Reference

A pointer or identifier referencing stored evidence used during decision construction or governance evaluation.

Evidence Representation

The format used to encode evidence artifacts for governance processing.

Evidence Retention

The policy-defined duration for which evidence artifacts must be preserved.

Evidence Reproducibility

The ability to reproduce or reconstruct evidence used in decision evaluation.

Evidence Source

The origin of a piece of evidence, such as a sensor, external system, or user input.

Evidence Type

A standardized category describing the origin or nature of evidence.

Baseline evidence types include:

SENSOR
USER_INPUT
STATE_SNAPSHOT
RETRIEVED_DOC
DERIVED_SIGNAL
EXTERNAL_EVENT

Executor

The actor responsible for carrying out an authorized decision.

Fail-Closed Behavior

A security principle requiring systems to deny execution when governance validation cannot be completed successfully.

Forward Compatibility

The ability of older implementations to safely process artifacts generated by newer protocol versions.

Governance Boundary

The mandatory interface between Decision Construction and Decision Execution where governance evaluation occurs.

Governance Boundary Integrity Test

A conformance test verifying that execution cannot occur without passing through the Governance Boundary.

Governance Outcome

The final authorization result produced by governance evaluation.

Canonical outcomes:

ALLOW
DENY
ESCALATE
DEFER
DEGRADE

Governor

The actor responsible for evaluating policies and producing governance outcomes.

Governed Execution

Execution of a decision after successful authorization at the Governance Boundary.

Human Review

A governance escalation mechanism requiring human approval prior to execution.

Incremental Adoption

A staged implementation approach in which governance capabilities are introduced progressively.

Integrity and Audit Block

The portion of a Decision Receipt responsible for integrity verification and audit metadata.

Interoperability

The ability of independent systems to exchange and validate Decision Receipts and governance artifacts.

Least Privilege

A security principle requiring actors to possess only the permissions necessary to perform their roles.

Major Version

A protocol version increment indicating incompatible changes.

Minor Version

A protocol version increment indicating backward-compatible enhancements.

Minimal Receipt (RETNA-A)

A minimal Decision Receipt structure required for basic governance compliance.

Observability-Only Adoption

An adoption stage in which Decision Receipts are generated for visibility and auditing without enforcing governance outcomes.

Outcome Block

The portion of a Decision Receipt recording the governance outcome and associated reason codes.

Payload Kind

A classification identifying the semantic category of a decision request used to route governance evaluation.

Policy

A set of enforceable rules evaluated at the Governance Boundary.

Policy Applicability

The scope within which a policy bundle is applied.

Policy Authority

The system or entity responsible for defining and distributing governance policies.

Policy Bundle

A collection of policy rules, constraints, and configuration parameters used during governance evaluation.

Policy Bundle Version

The version identifier associated with a policy bundle.

Policy Constraint

An individual rule within a policy restricting execution behavior.

Policy Evaluation

The process of applying policy rules and constraints to a candidate decision.

Policy Evaluation Block

The section of a Decision Receipt documenting policy evaluation results.

Privileged Execution

Execution of a decision that alters external state and therefore requires explicit authorization.

Producer

The actor responsible for constructing a candidate decision.

Protocol Version

The version identifier associated with a specific revision of the Protocol.

Reason Code

A standardized machine-readable code representing the result of policy evaluation.

Receipt Compatibility

The ability of systems to interpret Decision Receipts generated by other implementations.

Receipt Emission

The generation of a Decision Receipt as part of governance evaluation.

Receipt Integrity

The property ensuring that a Decision Receipt cannot be altered without detection.

Receipt Immutability

The property that once issued, a Decision Receipt cannot be modified.

Receipt Store

A system responsible for persisting Decision Receipts for audit and verification.

Retention Classification

A policy-defined category specifying how long governance artifacts must be retained.

Risk Class

A classification representing the potential impact or severity of a decision.

Separation of Duties

A governance principle requiring different actors to perform decision construction, evaluation, and execution.

Stability Guarantees

Protocol assurances ensuring that implementations remain interoperable across version updates.

Tamper-Evident Storage

A storage mechanism that detects unauthorized modification of governance artifacts.

Vendor Extension

An optional protocol extension introduced by a specific implementation.

Vendor-Neutral

A design principle ensuring the protocol can be implemented across independent systems without vendor lock-in.

Appendix B — Payload Kind Registry (Informative Reference)

This appendix points to the machine-readable payload kind registry used by the reference implementation.

Suggested publication note:

The reference implementation publishes a machine-readable payload kind registry describing canonical payload categories, privilege-sensitive categories, and baseline semantics. This registry is informative for v0.1 and does not override the normative rules of Sections 8 and 14.

Download the recommended companion artifact:
`retna_payload_kind_registry.json`

Appendix C — Canonical Reason Code Registry (Informative Reference)

This appendix points to the machine-readable canonical reason code registry used by the reference implementation.

Suggested publication note:

The reference implementation publishes a machine-readable reason code registry describing canonical baseline reason codes, category groupings, and short semantic descriptions. This registry is informative for v0.1 and does not override the normative rules of Section 9.

Download the recommended companion artifact:
`retna_reason_code_registry.json`