Internet-Draft
Intended Status: Informational
Expires: October 28, 2026

L. J. Reilly
Independent
April 28, 2026

Reilly Model Routing Protocol (RMRP): A Framework for Policy-Governed, Auditable AI Model Routing

draft-reilly-rmrp-00

Abstract

This document specifies the Reilly Model Routing Protocol (RMRP), a framework for policy-governed, auditable routing of inference requests across heterogeneous artificial intelligence (AI) model environments. RMRP defines the structural metadata, routing policy declaration, execution semantics, audit trail requirements, and cost attribution mechanisms necessary to govern how inference requests are directed to AI models in multi-model deployments.

The protocol is AI-provider agnostic and operates independently of any specific model architecture, inference runtime, vendor implementation, or transport layer. RMRP addresses the absence of a standardized protocol-layer specification governing how routing decisions are declared, transmitted, logged, and enforced across AI model deployments at organizational scale.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 28, 2026.

Copyright Notice

Table of Contents

1. Introduction 4
  1.1. Motivation 4
  1.2. Scope 5
  1.3. Design Principles 5
2. Terminology 6
3. RMRP Architecture Overview 8
  3.1. Architectural Layers 8
  3.2. Component Roles 9
  3.3. Request Lifecycle 10
4. Model Routing Decision (MRD) 11
  4.1. MRD Structure 11
  4.2. MRD Field Definitions 12
  4.3. Task Classification 15
  4.4. Model Tier Definitions 16
  4.5. Complexity Scoring 17
  4.6. MRD Example 18
5. Routing Policy Document (RPD) 19
  5.1. RPD Structure 19
  5.2. RPD Field Definitions 20
  5.3. Rule Evaluation Order 23
  5.4. Fallback Behavior 23
  5.5. RPD Example 24
6. Routing Execution Semantics 26
  6.1. Pre-Routing Validation 26
  6.2. Policy Resolution 26
  6.3. Model Selection 27
  6.4. Request Dispatch 27
  6.5. Response Handling 28
  6.6. Error and Fallback Handling 28
7. Audit Trail Requirements 30
  7.1. Audit Log Record (ALR) Structure 30
  7.2. ALR Field Definitions 31
  7.3. Audit Level Classes 33
  7.4. Retention Requirements 33
  7.5. ALR Example 34
8. Cost Attribution Framework 35
  8.1. Cost Attribution Record (CAR) 35
  8.2. CAR Field Definitions 36
  8.3. Budget Authority Chain 37
  8.4. Cost Ceiling Enforcement 38
  8.5. CAR Example 38
9. Governance and Authorization 39
  9.1. Policy Authority Model 39
  9.2. Policy Issuance and Signing 40
  9.3. Policy Versioning 41
  9.4. Override Mechanisms 41
10. Transport Considerations 42
  10.1. HTTP Transport 42
  10.2. Header Propagation 43
  10.3. Non-HTTP Transports 43
11. Security Considerations 44
  11.1. Policy Integrity 44
  11.2. MRD Tampering 44
  11.3. Audit Log Integrity 45
  11.4. Denial of Service 45
  11.5. Credential Exposure 45
12. Privacy Considerations 46
13. IANA Considerations 47
14. References 48
  14.1. Normative References 48
  14.2. Informative References 49
Acknowledgments 50
Author's Address 50

1. Introduction

1.1. Motivation

The deployment of artificial intelligence systems at organizational scale increasingly involves multiple AI models operating in parallel or in sequence. A given application may call upon lightweight models for classification tasks, mid-tier models for summarization, and advanced models for complex multi-step reasoning, all within a single request pipeline. This operational pattern is referred to throughout this document as multi-model deployment.

Despite the rapid proliferation of multi-model AI deployments, no standardized protocol exists that specifies how routing decisions between models should be declared, communicated, executed, audited, or governed. Current industry practice is characterized by ad hoc engineering decisions embedded in application code, vendor-specific gateway configurations, or informal internal policies with no interoperable representation.

This absence of a protocol standard produces several systemic problems:

- Routing logic is opaque and non-portable across systems and vendors.
- Cost attribution for model usage cannot be reliably traced to the policy decision that produced the expenditure.
- Audit records, where they exist, are inconsistent, incomplete, and not interoperable.
- Governance over who may define, modify, or override routing policy is informal and unenforceable at the protocol layer.
- Failure modes, escalation paths, and fallback behavior are undefined in any portable specification.

The Reilly Model Routing Protocol (RMRP) addresses each of these deficiencies by defining a provider-agnostic, transport-agnostic protocol framework for AI model routing governance.

1.2. Scope

This document specifies:

- The Model Routing Decision (MRD): a structured metadata object that accompanies every routed inference request, carrying the routing decision and its full governance context.
- The Routing Policy Document (RPD): a declarative specification that defines the rules by which routing decisions are made.
- Routing Execution Semantics: the normative procedures by which an RMRP-compliant router resolves, applies, and validates routing policy against an incoming request.
- The Audit Log Record (ALR): a standardized record structure for capturing each routing event for compliance, debugging, and accountability purposes.
- The Cost Attribution Record (CAR): a structure that traces the financial cost of each routed request to the policy, cost center, and budget authority that authorized it.

- o A Governance and Authorization model that defines how routing policy is issued, signed, versioned, and enforced.

This document does not specify the internal architecture of any AI model, the machine learning logic used to assess request complexity, the commercial pricing structure of any AI provider, or any application-layer inference API. RMRP is a governance and metadata protocol layer that sits above any such systems.

1.3. Design Principles

RMRP is designed according to the following principles:

Provider Agnosticism: RMRP MUST NOT assume the use of any specific AI model provider, vendor API, or proprietary infrastructure. All provider-specific identifiers are treated as opaque strings within the protocol.

Transport Agnosticism: RMRP metadata structures are defined as JSON objects. They may be transmitted over HTTP, message queues, RPC frameworks, or any other transport that supports structured data payloads.

Auditability by Default: Every routing decision produces an auditable record. Audit logging is not optional for conformant implementations.

Policy as a First-Class Object: Routing policy is a declared, versioned, signed artifact. Inline or implicit routing logic embedded in application code does not satisfy RMRP conformance.

Cost Traceability: Every routed request MUST be attributable to a cost center and budget authority. Unattributed inference costs are a conformance violation.

Least-Cost Sufficiency: Routing policy SHOULD direct requests to the least capable model tier sufficient to satisfy the task requirements. Routing to a higher tier MUST be justified by policy conditions.

Separation of Concerns: The routing decision layer is separate from the inference execution layer. An RMRP-compliant router makes and records a routing decision; it does not implement inference.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined for use in this document:

AI Model: A software system that accepts a structured input (referred to herein as an inference request) and produces a structured output (referred to herein as an inference response) using a learned computational process. This definition is intentionally broad and encompasses large language models, multimodal models, embedding models, classification models, and other learned inference systems.

Budget Authority: The organizational entity, role, or system identity that has been designated as responsible for approving

inference expenditure within a defined cost center. Budget Authority is referenced by identifier within RMRP structures and resolved externally to the protocol.

Complexity Score: A normalized floating-point value in the range [0.0, 1.0] that represents an assessment of the computational or semantic difficulty of an inference request relative to a defined task type. The method of computing the Complexity Score is outside the scope of this specification; RMRP defines only how this value is represented and used in routing decisions.

Conformant Router: An RMRP Routing Engine that implements all REQUIRED behaviors specified in this document.

Cost Attribution Record (CAR): A structured JSON object produced by a Conformant Router upon completion of a routed inference request, recording the financial cost of the request and attributing it to the applicable cost center, policy, and Budget Authority.

Cost Center: An organizational unit, project, team, application, or other logical grouping to which the financial cost of inference requests is attributed.

Fallback Model: The AI model or model tier to which a routing request is directed when the primary selected model is unavailable, returns an error, or exceeds a defined constraint.

Inference Request: A structured input submitted to an AI model for processing. The content and format of the inference request are outside the scope of this specification.

Inference Response: The structured output produced by an AI model in response to an Inference Request. The content and format of the inference response are outside the scope of this specification.

Model Identifier: An opaque string that uniquely identifies a specific AI model or model version within the scope of a deployment. Model Identifiers are defined and managed externally to RMRP.

Model Registry: An external system or configuration artifact that maps Model Identifiers to model capabilities, tier assignments, and endpoint information. RMRP does not specify the implementation of a Model Registry but requires that a Conformant Router have access to one.

Model Routing Decision (MRD): A structured JSON object produced by a Conformant Router that records the routing decision made for a specific inference request, including the selected model, the policy applied, and the full governance context.

Model Tier: A categorical classification of AI models according to their relative capability and cost. This specification defines three normative tiers: LIGHT, STANDARD, and ADVANCED. Implementations MAY define additional tiers subject to the constraints in Section 4.4.

Policy Authority: The organizational entity or role that has been granted the right to issue, sign, and publish Routing Policy Documents within a defined scope.

Routing Engine: The software component responsible for receiving an inference request, evaluating applicable Routing Policy Documents, producing a Model Routing Decision, and dispatching

the request to the selected model.

Routing Policy Document (RPD): A structured, versioned, signed JSON document that declares the rules by which a Routing Engine selects a target model for a given inference request.

Task Type: A categorical label that describes the nature of an inference request at the application level. RMRP defines a normative set of Task Types in Section 4.3. Implementations MAY extend this set using the extension mechanism defined therein.

Audit Log Record (ALR): A structured JSON object produced by a Conformant Router for each routing event, capturing the full decision trace, outcome, and timing of the routing operation.

RMRP Version: The protocol version string identifying the version of this specification to which an MRD, RPD, ALR, or CAR conforms. The version string for this specification is "1.0".

3. RMRP Architecture Overview

3.1. Architectural Layers

RMRP defines a governance layer that operates between the application layer and the AI model inference layer. This layer is not a transport protocol and does not replace any existing network or application protocol. It defines the metadata, policy, and audit structures that govern routing decisions.

The RMRP architectural layers are as follows:

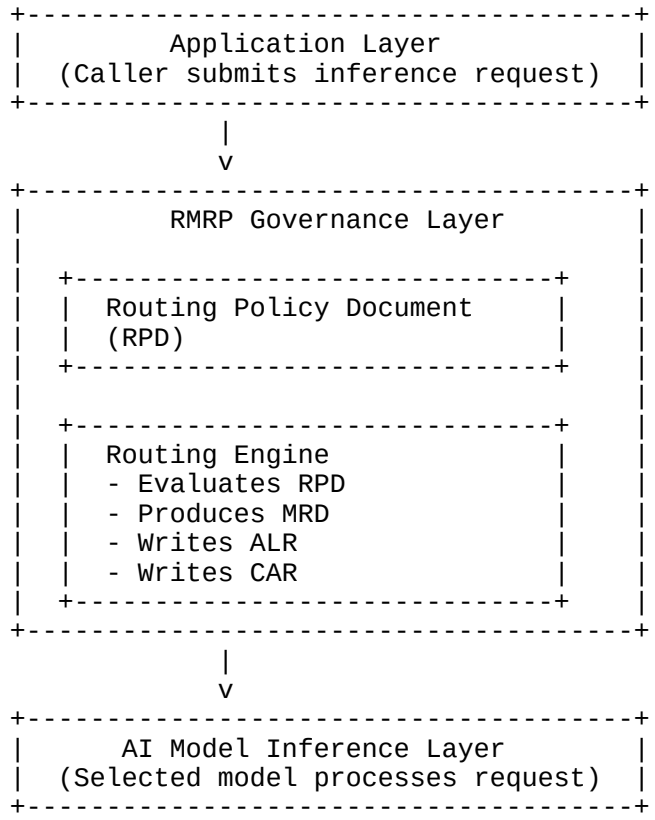

Figure 1: RMRP Architectural Layers

3.2. Component Roles

Policy Authority: Issues, signs, and publishes RPDs. The Policy

Authority is responsible for ensuring that RPDs reflect organizational cost governance, capability requirements, and compliance constraints. A Policy Authority MUST NOT be the same entity as the Routing Engine in deployments where separation of concerns is required by organizational policy, though this specification does not mandate such separation in all contexts.

Routing Engine: Receives inference requests from the application layer, resolves the applicable RPD, computes a routing decision, produces an MRD, dispatches the request to the selected model, and writes an ALR and CAR upon completion. A Conformant Router MUST perform all of these functions.

Model Registry: An external system consulted by the Routing Engine to resolve Model Identifiers to endpoints and validate tier assignments. RMRP does not specify the implementation of the Model Registry.

Audit Store: The persistent storage system into which the Routing Engine writes ALRs and CARs. The Audit Store MUST be write-once or append-only with respect to routing records to preserve audit integrity. Implementations MAY use cryptographic mechanisms to further ensure record immutability.

Budget Authority: Receives cost attribution data via CARs. Budget Authority systems are external to RMRP and consume CAR records for financial reporting and budget enforcement purposes.

3.3. Request Lifecycle

The lifecycle of an inference request under RMRP is as follows:

1. The application layer submits an inference request to the Routing Engine, optionally including request metadata such as Task Type, priority class, and cost center identifier.

2. The Routing Engine performs pre-routing validation as specified in Section 6.1.

3. The Routing Engine resolves the applicable RPD as specified in Section 6.2.

4. The Routing Engine evaluates RPD rules against the request metadata and computes a Complexity Score to select a target model tier and Model Identifier.

5. The Routing Engine produces a Model Routing Decision (MRD) documenting the decision and its full governance context.

6. The Routing Engine dispatches the inference request to the selected model, attaching the MRD as specified in Section 6.4.

7. The selected model processes the request and returns an inference response.

8. The Routing Engine receives the response, records actual token consumption and latency, and updates the ALR and CAR with outcome data.

9. The Routing Engine writes the completed ALR and CAR to the Audit Store.

10. The inference response is returned to the application layer, accompanied by a reference to the MRD for correlation purposes.

4. Model Routing Decision (MRD)

4.1. MRD Structure

The Model Routing Decision is a JSON object [RFC8259] that MUST be produced by a Conformant Router for every inference request processed. The MRD captures the routing decision and its full governance context in a portable, inspectable form.

The MRD MUST contain all REQUIRED fields defined in Section 4.2. OPTIONAL fields SHOULD be included when the relevant information is available to the Routing Engine. Additional fields not defined in this specification MAY be included using the extension mechanism defined in Section 4.2.

All field names are case-sensitive. All string values are UTF-8 encoded [RFC3629]. All timestamp values are ISO 8601 formatted strings in UTC with millisecond precision.

4.2. MRD Field Definitions

rmrp_version (string, REQUIRED)
: The RMRP protocol version string. For documents conforming to this specification, the value MUST be "1.0".

mrd_id (string, REQUIRED)
: A universally unique identifier for this MRD instance, formatted as a UUID [RFC9562]. This identifier is used to correlate the MRD with associated ALRs, CARs, and application logs.

request_id (string, REQUIRED)
: An identifier for the inference request as assigned by the caller. If the caller does not supply a request identifier, the Routing Engine MUST generate one and return it to the caller. Format is implementation-defined but MUST be unique within the scope of the deployment.

timestamp (string, REQUIRED)
: The UTC timestamp at which the Routing Engine produced this MRD. Format: ISO 8601 with millisecond precision. Example: "2026-04-28T17:00:00.000Z"

routing_policy_id (string, REQUIRED)
: The unique identifier of the Routing Policy Document applied to produce this routing decision. This value MUST correspond to the "policy_id" field of the applicable RPD.

routing_policy_version (string, REQUIRED)
: The version string of the RPD applied. This value MUST correspond to the "policy_version" field of the applicable RPD.

source_system (string, REQUIRED)
: An identifier for the application or system component that submitted the inference request. Format is implementation-defined. This field is used for attribution, auditing, and cost allocation.

task_type (string, REQUIRED)
: The Task Type classification of the inference request. MUST be one of the normative Task Type values defined in Section 4.3, or an extended value registered per the extension mechanism in Section 4.3.

complexity_score (number, REQUIRED)
: A floating-point value in the range [0.0, 1.0] representing

the assessed complexity of the inference request. The method of computation is outside the scope of this specification. A value of 0.0 represents the minimum assessed complexity for the given task type; a value of 1.0 represents the maximum.

selected_model_id (string, REQUIRED)
The Model Identifier of the AI model selected to process this inference request. This value MUST resolve to a registered model in the Model Registry.

selected_model_tier (string, REQUIRED)
The model tier assignment of the selected model. MUST be one of the normative tier values defined in Section 4.4.

routing_rationale (string, REQUIRED)
A human-readable description of the routing decision, identifying which RPD rule was matched and why the selected model tier was chosen. This field is intended for audit inspection and operational debugging.

cost_center (string, REQUIRED)
The identifier of the cost center to which the financial cost of this inference request is attributed. This value MUST correspond to a valid cost center identifier in the organization's Cost Attribution framework.

budget_authority_id (string, REQUIRED)
The identifier of the Budget Authority that approved inference expenditure for this cost center under the applicable RPD.

max_token_budget (integer, REQUIRED)
The maximum number of tokens (input plus output) authorized for this inference request under the applicable RPD rule. A value of -1 indicates no token ceiling is enforced by policy for this request. Routing Engines MUST NOT route requests where the estimated token consumption exceeds this value without triggering the fallback behavior defined in Section 6.6.

priority_class (string, REQUIRED)
The priority classification of the inference request. MUST be one of: "CRITICAL", "HIGH", "STANDARD", "BATCH". Priority class MAY influence model selection and queuing behavior. See Section 4.5 for priority class semantics.

fallback_model_id (string, OPTIONAL)
The Model Identifier of the fallback model to be used if the selected model is unavailable or returns an error. If present, this value MUST resolve to a registered model in the Model Registry.

fallback_model_tier (string, OPTIONAL)
The model tier assignment of the fallback model, if specified. MUST be one of the normative tier values defined in Section 4.4 if present.

chain_id (string, OPTIONAL)
An identifier linking this inference request to a broader multi-step request pipeline or agent chain. Used for correlating multiple MRDs produced within a single logical workflow.

chain_step (integer, OPTIONAL)
The ordinal position of this inference request within the chain identified by "chain_id". MUST be a non-negative integer. The first step in a chain is 0.

estimated_input_tokens (integer, OPTIONAL)
The estimated number of input tokens for this inference request, as assessed by the Routing Engine prior to dispatch.

estimated_output_tokens (integer, OPTIONAL)
The estimated number of output tokens for this inference request, as assessed by the Routing Engine prior to dispatch.

audit_level (string, REQUIRED)
The audit level class applied to this routing event. MUST be one of the normative audit level values defined in Section 7.3.

extensions (object, OPTIONAL)
A JSON object containing implementation-specific or deployment-specific fields not defined in this specification. Extension field names MUST use a reverse-DNS prefix to avoid collisions (e.g., "com.example.custom_field"). The presence of extension fields MUST NOT alter the interpretation of any normative field defined in this specification.

4.3. Task Classification

RMRP defines the following normative Task Types. These values are case-sensitive and MUST be used verbatim in the "task_type" field of the MRD and in RPD rule conditions.

CLASSIFICATION A request whose primary output is a categorical label or score applied to input content. Includes sentiment analysis, intent detection, content moderation, and similar tasks. Typically low complexity.

EXTRACTION A request whose primary output is structured data extracted from unstructured input, including named entity recognition, key-value extraction, and table parsing.

SUMMARIZATION A request whose primary output is a condensed representation of a larger input document or corpus.

GENERATION A request whose primary output is novel content generated in response to a prompt, including text generation, code generation, and creative writing tasks.

REASONING A request that requires multi-step logical inference, mathematical computation, or structured problem-solving. Typically high complexity.

EMBEDDING A request whose primary output is a vector representation of the input content. Embedding requests SHOULD be routed to models optimized for embedding generation.

RETRIEVAL A request that involves retrieval-augmented generation or query-driven document retrieval. Complexity is a function of retrieval corpus size and query ambiguity.

TRANSFORMATION A request whose primary output is a transformed version of the input (e.g., translation, reformatting, normalization, or style transfer).

AGENTIC A request submitted within an autonomous agent pipeline that may produce tool calls, multi-turn interactions, or sub-task decomposition. Agentic requests SHOULD be assigned higher complexity scores by default given their potential for recursive resource consumption.

MULTIMODAL A request that includes non-text input modalities such as images, audio, or video, in addition to or in place of

text input.

Implementations MAY define additional Task Types using the "extensions" mechanism. Extended Task Type values MUST use a reverse-DNS prefix (e.g., "com.example.CUSTOM_TASK"). RPD rules that reference extended Task Types MUST be ignored by Routing Engines that do not recognize the extended value, and fallback behavior as defined in Section 6.6 MUST be applied.

4.4. Model Tier Definitions

RMRP defines three normative model tiers. Tier assignment is the responsibility of the operator and is recorded in the Model Registry. RMRP does not prescribe which specific models belong to which tier; this is a deployment-time configuration decision.

LIGHT Models in the LIGHT tier are optimized for low-latency, high-throughput processing of tasks with low-to-moderate complexity. LIGHT tier models are expected to be the lowest-cost option in a deployment. LIGHT tier SHOULD be the default routing target for CLASSIFICATION, EXTRACTION, EMBEDDING, and TRANSFORMATION task types unless policy conditions require escalation.

STANDARD Models in the STANDARD tier provide a balanced capability-to-cost profile. STANDARD tier is appropriate for SUMMARIZATION, GENERATION, and RETRIEVAL tasks at moderate complexity scores, and for REASONING tasks at low complexity scores.

ADVANCED Models in the ADVANCED tier provide maximum available capability for high-complexity tasks. ADVANCED tier MUST only be selected when policy conditions explicitly authorize it and the request complexity or task type requires capabilities unavailable in lower tiers. Routing to ADVANCED tier without explicit RPD authorization is a conformance violation.

Implementations MAY define additional tiers using the "extensions" mechanism. Extended tier values MUST NOT replace or supersede the normative tier definitions above.

4.5. Complexity Scoring

The Complexity Score is a normalized floating-point value in [0.0, 1.0] that the Routing Engine assigns to each inference request prior to RPD rule evaluation. RMRP does not mandate a specific algorithm for computing the Complexity Score. Conformant implementations MUST document the method used to produce this value for audit purposes.

Informative guidance for Complexity Score computation includes:

- Input token count relative to model context window capacity.
- Presence of multi-step instructions or chained subtasks.
- Ambiguity of the input as assessed by a lightweight classifier.
- Historical accuracy of lower-tier models on similar inputs.
- Structural complexity indicators such as nested conditionals, mathematical expressions, or code with high cyclomatic complexity.

Priority class semantics are as follows. Note: the value "STANDARD" used for priority class is distinct from the model tier "STANDARD" defined in Section 4.4. These identifiers exist in separate namespaces within the protocol and MUST NOT be conflated.

CRITICAL Requests that require immediate processing. Priority class CRITICAL MUST NOT be routed to BATCH processing queues. CRITICAL requests MAY bypass certain cost ceiling constraints as defined in the applicable RPD.

HIGH Requests that require low-latency processing but are not operationally critical.

STANDARD Default priority for interactive workloads.

BATCH Requests that are tolerant of high latency in exchange for reduced per-token cost. BATCH requests SHOULD be queued for asynchronous processing where the model provider supports it.

4.6. MRD Example

The following is a non-normative example of a conformant MRD:

```
{
  "rmrp_version": "1.0",
  "mrd_id": "550e8400-e29b-41d4-a716-446655440000",
  "request_id": "req-20260428-00192",
  "timestamp": "2026-04-28T17:00:00.000Z",
  "routing_policy_id": "rpd-prod-engineering-v3",
  "routing_policy_version": "3.2.1",
  "source_system": "api-gateway.internal",
  "task_type": "REASONING",
  "complexity_score": 0.82,
  "selected_model_id": "provider-alpha/model-advanced-v2",
  "selected_model_tier": "ADVANCED",
  "routing_rationale": "Rule R-07 matched: task_type=REASONING,
    complexity_score 0.82 exceeds STANDARD tier threshold 0.75.
    ADVANCED tier authorized by policy for cost_center=eng-ai.",
  "cost_center": "eng-ai",
  "budget_authority_id": "ba-vp-engineering-001",
  "max_token_budget": 8192,
  "priority_class": "HIGH",
  "fallback_model_id": "provider-alpha/model-standard-v4",
  "fallback_model_tier": "STANDARD",
  "chain_id": "chain-pipeline-20260428-00041",
  "chain_step": 2,
  "estimated_input_tokens": 2048,
  "estimated_output_tokens": 1024,
  "audit_level": "FULL",
  "extensions": {}
}
```

5. Routing Policy Document (RPD)

5.1. RPD Structure

The Routing Policy Document is a structured, versioned JSON document that declares the rules by which a Routing Engine selects a target model for a given inference request. A conformant RPD MUST be a valid JSON object [RFC8259].

RPDs MUST be digitally signed by the issuing Policy Authority using a mechanism that allows the Routing Engine to verify authenticity and detect tampering. This specification RECOMMENDS the use of JSON Web Signatures (JWS) as defined in [RFC7515].

An RPD MUST be version-controlled. Routing Engines MUST record the specific RPD version applied to each routing decision in the MRD. Superseded RPD versions MUST be retained in the Audit Store for the retention period defined in Section 7.4.

5.2. RPD Field Definitions

rmrp_version (string, REQUIRED)
The RMRP protocol version string. MUST be "1.0" for this specification.

policy_id (string, REQUIRED)
A unique identifier for this Routing Policy Document within the deployment scope. Policy IDs MUST be stable across versions of the same policy; different versions of the same policy MUST share the same "policy_id".

policy_version (string, REQUIRED)
A semantic version string [semver] identifying this version of the policy. Format: MAJOR.MINOR.PATCH. A change to routing logic MUST increment MINOR or MAJOR. A change to metadata only MAY increment PATCH.

policy_name (string, REQUIRED)
A human-readable name for this policy, suitable for display in audit interfaces.

policy_authority_id (string, REQUIRED)
The identifier of the Policy Authority that issued this document.

effective_date (string, REQUIRED)
The UTC timestamp from which this policy version is effective. Routing Engines MUST NOT apply a policy version prior to its effective date.

expiration_date (string, OPTIONAL)
The UTC timestamp after which this policy version is no longer valid. If present, Routing Engines MUST NOT apply this policy version after the expiration date and MUST trigger the fallback behavior defined in Section 5.4.

scope (object, REQUIRED)
Defines the set of source systems, cost centers, and task types to which this policy applies.

scope.source_systems (array of strings, OPTIONAL)
If present, this policy applies only to inference requests originating from the listed source system identifiers. If absent, the policy applies to all source systems unless overridden by a more specific policy.

scope.cost_centers (array of strings, OPTIONAL)
If present, this policy applies only to inference requests attributed to the listed cost center identifiers.

scope.task_types (array of strings, OPTIONAL)
If present, this policy applies only to inference requests of the listed task types.

default_rule (object, REQUIRED)
The routing rule applied when no other rule in the "rules" array produces a match. The default rule MUST specify at minimum a "target_tier" and a "max_token_budget". The default rule MUST NOT specify conditions.

rules (array of objects, REQUIRED)
An ordered array of routing rules. MUST contain at least one rule. Rules MUST be evaluated in array order. The first rule whose conditions are satisfied by the inference request MUST

be applied. Subsequent rules MUST NOT be evaluated after a match.

Each rule object contains the following fields:

rule_id (string, REQUIRED)
: A unique identifier for this rule within the RPD. Rule IDs MUST be stable across policy versions.

rule_description (string, OPTIONAL)
: A human-readable description of the rule's intent.

conditions (object, REQUIRED for non-default rules)
: A JSON object specifying the conditions under which this rule applies. All specified conditions MUST be satisfied for the rule to match (logical AND). If no conditions are specified, the rule matches all requests (and SHOULD only appear as the default rule).

  conditions.task_types (array of strings, OPTIONAL)
  : The rule matches only if the request "task_type" is one of the listed values.

  conditions.complexity_min (number, OPTIONAL)
  : The rule matches only if the request "complexity_score" is greater than or equal to this value.

  conditions.complexity_max (number, OPTIONAL)
  : The rule matches only if the request "complexity_score" is less than this value.

  conditions.priority_classes (array of strings, OPTIONAL)
  : The rule matches only if the request "priority_class" is one of the listed values.

  conditions.source_systems (array of strings, OPTIONAL)
  : The rule matches only if the request "source_system" is one of the listed values.

  conditions.cost_centers (array of strings, OPTIONAL)
  : The rule matches only if the request "cost_center" is one of the listed values.

  conditions.chain_step_max (integer, OPTIONAL)
  : The rule matches only if the request "chain_step" is less than or equal to this value. Used to constrain model tier selection in early pipeline steps.

target_tier (string, REQUIRED)
: The model tier to which matching requests are routed. MUST be one of the normative tier values defined in Section 4.4.

target_model_id (string, OPTIONAL)
: If present, the Routing Engine MUST route matching requests to this specific model, subject to availability. If the specified model is unavailable, fallback behavior applies.

fallback_tier (string, OPTIONAL)
: The model tier to which the request is routed if the primary target model is unavailable. If absent, the Routing Engine MUST use the "default_rule" target as fallback.

fallback_model_id (string, OPTIONAL)
: If present, the specific fallback model identifier. Evaluated after "fallback_tier".

max_token_budget (integer, REQUIRED)
The maximum total tokens (input plus output) authorized for requests matching this rule. A value of -1 indicates no ceiling is enforced by this rule. Routing Engines MUST enforce this constraint before dispatch.

cost_ceiling_usd (number, OPTIONAL)
The maximum estimated cost in USD authorized for a single inference request matching this rule. If present, the Routing Engine MUST reject or reroute requests whose estimated cost exceeds this value. Estimation method is implementation-defined and MUST be documented.

audit_level (string, REQUIRED)
The audit level class applied to routing events matching this rule. MUST be one of the normative audit level values defined in Section 7.3.

allow_advanced_escalation (boolean, OPTIONAL)
If true, and if the "target_tier" is STANDARD, the Routing Engine MAY escalate to ADVANCED tier if the complexity score exceeds the escalation_threshold. Default: false.

escalation_threshold (number, OPTIONAL)
The complexity score threshold above which escalation to ADVANCED tier is permitted when "allow_advanced_escalation" is true. MUST be in [0.0, 1.0].

5.3. Rule Evaluation Order

A Conformant Router MUST evaluate RPD rules in the following order:

1. Filter out rules whose conditions do not match the request metadata as described in Section 5.2.

2. Apply the first matching rule in array order.

3. If no rule matches, apply the "default_rule".

4. If the "default_rule" is absent or invalid, the Routing Engine MUST reject the request and write an ALR with outcome "POLICY_ERROR".

5.4. Fallback Behavior

The following conditions MUST trigger fallback behavior:

- The selected model returns an HTTP 5xx error or equivalent transport-level failure.

- The selected model is not resolvable in the Model Registry.

- The estimated token count exceeds "max_token_budget".

- The estimated cost exceeds "cost_ceiling_usd" (if present).

- The RPD "expiration_date" has passed.

When fallback is triggered, the Routing Engine MUST:

1. Attempt routing to the "fallback_model_id" (if specified) or the model at "fallback_tier".

2. Record the fallback event in the ALR with the original selection, the fallback target, and the reason for fallback.

3. If the fallback model also fails, the Routing Engine MUST return an error to the caller and write an ALR with outcome "ROUTING_FAILURE".

5.5. RPD Example

The following is a non-normative example of a conformant RPD:

```
{
  "rmrp_version": "1.0",
  "policy_id": "rpd-prod-engineering-v3",
  "policy_version": "3.2.1",
  "policy_name": "Engineering Production AI Routing Policy",
  "policy_authority_id": "pa-cto-office-001",
  "effective_date": "2026-04-01T00:00:00.000Z",
  "expiration_date": "2026-10-01T00:00:00.000Z",
  "scope": {
    "source_systems": ["api-gateway.internal", "agent-runner.internal"],
    "cost_centers": ["eng-ai", "eng-platform"],
    "task_types": null
  },
  "default_rule": {
    "target_tier": "LIGHT",
    "max_token_budget": 2048,
    "audit_level": "STANDARD"
  },
  "rules": [
    {
      "rule_id": "R-01",
      "rule_description": "Batch embedding requests to LIGHT tier.",
      "conditions": {
        "task_types": ["EMBEDDING"],
        "priority_classes": ["BATCH"]
      },
      "target_tier": "LIGHT",
      "max_token_budget": 4096,
      "audit_level": "MINIMAL"
    },
    {
      "rule_id": "R-02",
      "rule_description": "Low-complexity classification to LIGHT.",
      "conditions": {
        "task_types": ["CLASSIFICATION", "EXTRACTION"],
        "complexity_max": 0.4
      },
      "target_tier": "LIGHT",
      "max_token_budget": 1024,
      "audit_level": "MINIMAL"
    },
    {
      "rule_id": "R-03",
      "rule_description": "Moderate generation to STANDARD tier.",
      "conditions": {
        "task_types": ["GENERATION", "SUMMARIZATION"],
        "complexity_min": 0.3,
        "complexity_max": 0.75
      },
      "target_tier": "STANDARD",
      "max_token_budget": 4096,
      "audit_level": "STANDARD"
    },
    {
      "rule_id": "R-04",
      "rule_description": "CRITICAL priority requests to STANDARD
        minimum.",
```

```
      "conditions": {
        "priority_classes": ["CRITICAL"]
      },
      "target_tier": "STANDARD",
      "fallback_tier": "ADVANCED",
      "max_token_budget": 8192,
      "audit_level": "FULL"
    },
    {
      "rule_id": "R-05",
      "rule_description": "High-complexity AGENTIC and REASONING
        requests to STANDARD with escalation permitted.",
      "conditions": {
        "task_types": ["AGENTIC", "REASONING"],
        "complexity_min": 0.5
      },
      "target_tier": "STANDARD",
      "allow_advanced_escalation": true,
      "escalation_threshold": 0.75,
      "max_token_budget": 16384,
      "cost_ceiling_usd": 0.50,
      "audit_level": "FULL"
    },
    {
      "rule_id": "R-06",
      "rule_description": "Multimodal requests to STANDARD tier.",
      "conditions": {
        "task_types": ["MULTIMODAL"]
      },
      "target_tier": "STANDARD",
      "max_token_budget": 8192,
      "audit_level": "STANDARD"
    },
    {
      "rule_id": "R-07",
      "rule_description": "High-complexity REASONING above
        threshold to ADVANCED.",
      "conditions": {
        "task_types": ["REASONING"],
        "complexity_min": 0.75
      },
      "target_tier": "ADVANCED",
      "fallback_tier": "STANDARD",
      "max_token_budget": 8192,
      "cost_ceiling_usd": 1.00,
      "audit_level": "FULL"
    }
  ]
}
```

6. Routing Execution Semantics

6.1. Pre-Routing Validation

Upon receipt of an inference request, a Conformant Router MUST perform the following validation steps before proceeding:

1. Verify that a valid RPD is available and has not expired. If no valid RPD is resolvable for the request context, the Routing Engine MUST reject the request with error code RMRP-001 (Policy Not Found).

2. Verify that the "source_system" identifier is present and recognized.

3. Verify that a "cost_center" is associated with the request, either supplied by the caller or resolvable from the "source_system" identifier via configuration.

4. Verify that the "budget_authority_id" associated with the cost center is active and has not been revoked.

5. Verify that the "task_type" is a recognized value per Section 4.3 or a registered extension value.

Requests that fail pre-routing validation MUST be rejected. Rejected requests MUST have an ALR written with outcome "VALIDATION_FAILURE" identifying which validation step failed.

6.2. Policy Resolution

The Routing Engine MUST resolve the applicable RPD using the following procedure:

1. Identify all RPDs whose "scope" matches the request (source_system, cost_center, task_type).

2. If multiple RPDs match, apply the most specific RPD as determined by the number of scope constraints satisfied.

3. If specificity is equal across multiple matching RPDs, apply the RPD with the most recent "effective_date".

4. Record the selected "policy_id" and "policy_version" in the MRD.

Implementations that maintain a single global RPD are not required to perform policy resolution but MUST still record the policy_id and policy_version in every MRD.

6.3. Model Selection

After RPD resolution, the Routing Engine MUST:

1. Compute or accept a Complexity Score for the request.

2. Evaluate RPD rules in order per Section 5.3.

3. Identify the matched rule and extract "target_tier" and, if present, "target_model_id".

4. If "target_model_id" is specified, resolve it in the Model Registry and verify it is available.

5. If "target_model_id" is absent, select an available model from the Model Registry whose tier matches "target_tier". Model selection within a tier is implementation-defined.

6. Evaluate "allow_advanced_escalation" and "escalation_threshold" if present.

7. Verify that the estimated token count does not exceed "max_token_budget".

8. Verify that the estimated cost does not exceed "cost_ceiling_usd" if present.

9. Produce and record the MRD.

6.4. Request Dispatch

The Routing Engine MUST dispatch the inference request to the selected model endpoint with the following requirements:

- The MRD MUST be attached to the dispatched request. In HTTP transport, this MUST be accomplished via the "RMRP-MRD" header or request body attachment per Section 10.1. In other transports, attachment is per Section 10.3.
- The "request_id" from the MRD MUST be forwarded to the model provider where the provider’s API supports a correlation identifier.
- All dispatch operations MUST be performed over encrypted transport (TLS 1.2 minimum, TLS 1.3 RECOMMENDED) per [RFC8446].

6.5. Response Handling

Upon receipt of an inference response, the Routing Engine MUST:

1. Record the actual input and output token counts from the response if available.
2. Record the end-to-end latency of the routing and inference operation.
3. Verify that actual token consumption did not exceed "max_token_budget". If it did, this MUST be recorded in the ALR as a budget overrun event.
4. Produce and finalize the ALR and CAR records.
5. Write completed ALR and CAR to the Audit Store.
6. Return the inference response to the caller with the "mrd_id" attached for correlation.

6.6. Error and Fallback Handling

Error codes defined by this specification:

RMRP-001 Policy Not Found. No valid RPD is resolvable for the request context.

RMRP-002 Validation Failure. The request failed pre-routing validation. Details MUST be included in the ALR.

RMRP-003 Budget Exceeded. The estimated or actual token or cost consumption exceeds authorized limits.

RMRP-004 Model Unavailable. The selected model is not reachable or returned a transport-level error.

RMRP-005 Fallback Exhausted. All fallback options have been attempted and failed.

RMRP-006 Policy Expired. The applicable RPD has passed its expiration date.

RMRP-007 Audit Store Failure. The Routing Engine was unable to write the ALR or CAR to the Audit Store. This is a CRITICAL error; the Routing Engine SHOULD halt request processing until Audit Store availability is restored.

All error events MUST result in an ALR record. Routing Engines

MUST NOT silently suppress routing errors.

7. Audit Trail Requirements

7.1. Audit Log Record (ALR) Structure

A Conformant Router MUST produce one Audit Log Record for each routing event. The ALR is a JSON object [RFC8259]. Each ALR MUST be written to the Audit Store before the inference response is returned to the caller.

ALRs MUST be immutable after writing. The Audit Store MUST be append-only or equivalent with respect to routing records. Implementations MAY use cryptographic hash chaining, blockchain anchoring, or other mechanisms to provide tamper-evidence for the ALR sequence.

7.2. ALR Field Definitions

rmrp_version (string, REQUIRED)
: RMRP protocol version. MUST be "1.0".

alr_id (string, REQUIRED)
: A UUID [RFC9562] uniquely identifying this ALR.

mrd_id (string, REQUIRED)
: The "mrd_id" of the MRD associated with this routing event.

request_id (string, REQUIRED)
: The "request_id" of the inference request.

timestamp_routing_start (string, REQUIRED)
: UTC timestamp at which the Routing Engine began processing the request.

timestamp_dispatch (string, REQUIRED)
: UTC timestamp at which the Routing Engine dispatched the request to the selected model.

timestamp_response (string, OPTIONAL)
: UTC timestamp at which the Routing Engine received the inference response. Absent if the request failed before a response was received.

timestamp_alr_written (string, REQUIRED)
: UTC timestamp at which the ALR was committed to the Audit Store.

routing_policy_id (string, REQUIRED)
: The "policy_id" of the RPD applied.

routing_policy_version (string, REQUIRED)
: The "policy_version" of the RPD applied.

matched_rule_id (string, REQUIRED)
: The "rule_id" of the RPD rule that matched this request. MUST be "default_rule" if the default rule was applied. MUST be absent or null if outcome is "VALIDATION_FAILURE" or "POLICY_ERROR".

source_system (string, REQUIRED)
: The source system identifier from the MRD.

task_type (string, REQUIRED)
: The task type from the MRD.

```
complexity_score  (number, REQUIRED)
   The complexity score from the MRD.

priority_class  (string, REQUIRED)
   The priority class from the MRD.

cost_center  (string, REQUIRED)
   The cost center from the MRD.

budget_authority_id  (string, REQUIRED)
   The budget authority from the MRD.

selected_model_id  (string, REQUIRED)
   The Model Identifier selected.

selected_model_tier  (string, REQUIRED)
   The model tier selected.

fallback_triggered  (boolean, REQUIRED)
   True if fallback routing was triggered during this event.

fallback_reason  (string, OPTIONAL)
   The reason fallback was triggered. REQUIRED if
   "fallback_triggered" is true.

fallback_model_id  (string, OPTIONAL)
   The Model Identifier used for fallback. REQUIRED if
   "fallback_triggered" is true.

outcome  (string, REQUIRED)
   The outcome of the routing event. MUST be one of:
   "SUCCESS", "FALLBACK_SUCCESS", "VALIDATION_FAILURE",
   "POLICY_ERROR", "ROUTING_FAILURE", "BUDGET_EXCEEDED",
   "POLICY_EXPIRED".

error_code  (string, OPTIONAL)
   The RMRP error code (e.g., "RMRP-004") if outcome is not
   "SUCCESS" or "FALLBACK_SUCCESS".

error_detail  (string, OPTIONAL)
   A human-readable description of the error.

actual_input_tokens  (integer, OPTIONAL)
   Actual input token count from the inference response.

actual_output_tokens  (integer, OPTIONAL)
   Actual output token count from the inference response.

actual_total_tokens  (integer, OPTIONAL)
   Sum of actual_input_tokens and actual_output_tokens.

budget_overrun  (boolean, REQUIRED)
   True if actual_total_tokens exceeded max_token_budget.

latency_routing_ms  (integer, OPTIONAL)
   Duration in milliseconds from routing start to dispatch.

latency_inference_ms  (integer, OPTIONAL)
   Duration in milliseconds from dispatch to response receipt.

latency_total_ms  (integer, OPTIONAL)
   Total duration in milliseconds from routing start to ALR
   write.

audit_level  (string, REQUIRED)
```

The audit level class applied to this event.

chain_id (string, OPTIONAL)
Chain identifier, if applicable.

chain_step (integer, OPTIONAL)
Chain step, if applicable.

previous_alr_id (string, OPTIONAL)
The "alr_id" of the immediately preceding ALR in the Audit Store. Used for hash chaining. RECOMMENDED for implementations that implement tamper-evident audit logs.

alr_hash (string, OPTIONAL)
A cryptographic hash of the canonical serialization of this ALR, computed prior to writing the "alr_hash" field itself. Hash algorithm MUST be identified in the "alr_hash_algorithm" field if present.

alr_hash_algorithm (string, OPTIONAL)
The hash algorithm used to compute "alr_hash". RECOMMENDED values: "SHA-256", "SHA3-512", "BLAKE3".

7.3. Audit Level Classes

MINIMAL Required for low-risk, high-volume routing events such as batch EMBEDDING tasks. ALR MUST include all REQUIRED fields. Token and latency fields are OPTIONAL.

STANDARD Default audit level for interactive workloads. ALR MUST include all REQUIRED fields and all timing fields.

FULL Required for ADVANCED tier routing, CRITICAL priority requests, high-cost requests, and any request where "allow_advanced_escalation" is true. ALR MUST include all defined fields. Implementations SHOULD compute and record "alr_hash" for FULL-level records.

7.4. Retention Requirements

ALRs and CARs MUST be retained for a minimum of 90 days. Implementations operating in regulated environments SHOULD retain records for a minimum of 7 years or the applicable regulatory retention period, whichever is longer.

Superseded RPD versions MUST be retained for the same period as the ALRs that reference them.

Audit Store implementations MUST support retrieval of ALRs by "mrd_id", "request_id", "chain_id", "cost_center", "routing_policy_id", and date range.

7.5. ALR Example

The following is a non-normative example of a conformant ALR:


```
{
  "rmrp_version": "1.0",
  "alr_id": "7f3b2c1a-0001-4d2e-9f8b-112233445566",
  "mrd_id": "550e8400-e29b-41d4-a716-446655440000",
  "request_id": "req-20260428-00192",
  "timestamp_routing_start": "2026-04-28T17:00:00.000Z",
  "timestamp_dispatch": "2026-04-28T17:00:00.032Z",
  "timestamp_response": "2026-04-28T17:00:02.187Z",
  "timestamp_alr_written": "2026-04-28T17:00:02.201Z",
  "routing_policy_id": "rpd-prod-engineering-v3",
```

```
  "routing_policy_version": "3.2.1",
  "matched_rule_id": "R-07",
  "source_system": "api-gateway.internal",
  "task_type": "REASONING",
  "complexity_score": 0.82,
  "priority_class": "HIGH",
  "cost_center": "eng-ai",
  "budget_authority_id": "ba-vp-engineering-001",
  "selected_model_id": "provider-alpha/model-advanced-v2",
  "selected_model_tier": "ADVANCED",
  "fallback_triggered": false,
  "outcome": "SUCCESS",
  "actual_input_tokens": 2041,
  "actual_output_tokens": 987,
  "actual_total_tokens": 3028,
  "budget_overrun": false,
  "latency_routing_ms": 32,
  "latency_inference_ms": 2155,
  "latency_total_ms": 2201,
  "audit_level": "FULL",
  "chain_id": "chain-pipeline-20260428-00041",
  "chain_step": 2,
  "alr_hash_algorithm": "SHA-256",
  "alr_hash": "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855"
}
```


8. Cost Attribution Framework

8.1. Cost Attribution Record (CAR)

A Conformant Router MUST produce one Cost Attribution Record for each routing event that results in an inference response, whether successful or via fallback. CARs MUST NOT be produced for requests that fail before dispatch.

The CAR is a JSON object [RFC8259]. CARs MUST be written to the Audit Store concurrently with or immediately following the associated ALR.

8.2. CAR Field Definitions

rmrp_version (string, REQUIRED)
: RMRP protocol version. MUST be "1.0".

car_id (string, REQUIRED)
: A UUID [RFC9562] uniquely identifying this CAR.

mrd_id (string, REQUIRED)
: The "mrd_id" of the associated MRD.

alr_id (string, REQUIRED)
: The "alr_id" of the associated ALR.

request_id (string, REQUIRED)
: The inference request identifier.

timestamp (string, REQUIRED)
: UTC timestamp of CAR production.

cost_center (string, REQUIRED)
: The cost center to which this expenditure is attributed.

budget_authority_id (string, REQUIRED)
: The Budget Authority identifier.

routing_policy_id (string, REQUIRED)
The RPD policy identifier.

routing_policy_version (string, REQUIRED)
The RPD policy version.

matched_rule_id (string, REQUIRED)
The rule that authorized this expenditure.

model_provider (string, OPTIONAL)
An opaque identifier for the AI model provider. This field is for organizational attribution and does not affect protocol behavior.

selected_model_id (string, REQUIRED)
The Model Identifier selected.

selected_model_tier (string, REQUIRED)
The model tier used for this request.

actual_input_tokens (integer, OPTIONAL)
Actual input token count.

actual_output_tokens (integer, OPTIONAL)
Actual output token count.

actual_total_tokens (integer, OPTIONAL)
Total actual token count.

estimated_cost_usd (number, OPTIONAL)
The estimated cost in USD at the time of routing, as computed by the Routing Engine. Estimation method is implementation-defined and MUST be documented.

actual_cost_usd (number, OPTIONAL)
The actual cost in USD as reported by the model provider or computed from actual token counts and known pricing.

cost_computation_method (string, OPTIONAL)
A description of the method used to compute cost figures. MUST be present if either "estimated_cost_usd" or "actual_cost_usd" is present.

authorized_cost_ceiling_usd (number, OPTIONAL)
The "cost_ceiling_usd" from the matched RPD rule, if any.

ceiling_exceeded (boolean, REQUIRED)
True if "actual_cost_usd" exceeds "authorized_cost_ceiling_usd". False if no ceiling was defined.

chain_id (string, OPTIONAL)
Chain identifier, if applicable.

chain_step (integer, OPTIONAL)
Chain step, if applicable.

8.3. Budget Authority Chain

The Budget Authority Chain is the traceable sequence of authorization that links an inference expenditure to the organizational entity responsible for it. In RMRP, this chain is represented implicitly through the combination of:

- The "cost_center" field, which identifies the organizational unit incurring the cost.

- The "budget_authority_id" field, which identifies the entity that approved inference expenditure for that cost center.

- The "routing_policy_id" and "routing_policy_version" fields, which identify the policy document that authorized the specific routing decision.

- The "policy_authority_id" field in the RPD, which identifies the entity that issued the policy.

External budget management systems consuming CAR records MUST be able to reconstruct the full authorization chain from these fields. RMRP does not specify the implementation of budget management systems.

8.4. Cost Ceiling Enforcement

When a "cost_ceiling_usd" is defined in the matched RPD rule, the Routing Engine MUST:

1. Compute or obtain an estimated cost for the request before dispatch.

2. Compare the estimated cost to the "cost_ceiling_usd".

3. If the estimated cost exceeds the ceiling, the Routing Engine MUST attempt to reroute to the "fallback_tier" or "fallback_model_id" as specified in Section 5.4.

4. If fallback also exceeds the ceiling, the Routing Engine MUST reject the request with error code RMRP-003 and write an ALR with outcome "BUDGET_EXCEEDED".

Cost ceiling enforcement based on estimated cost is a pre-dispatch control. Post-dispatch overruns MUST be recorded in the CAR as "ceiling_exceeded: true" but do not retroactively fail the completed request.

8.5. CAR Example

The following is a non-normative example of a conformant CAR:

```
{
  "rmrp_version": "1.0",
  "car_id": "ab12cd34-5678-4ef0-9012-abcdef012345",
  "mrd_id": "550e8400-e29b-41d4-a716-446655440000",
  "alr_id": "7f3b2c1a-0001-4d2e-9f8b-112233445566",
  "request_id": "req-20260428-00192",
  "timestamp": "2026-04-28T17:00:02.205Z",
  "cost_center": "eng-ai",
  "budget_authority_id": "ba-vp-engineering-001",
  "routing_policy_id": "rpd-prod-engineering-v3",
  "routing_policy_version": "3.2.1",
  "matched_rule_id": "R-07",
  "model_provider": "provider-alpha",
  "selected_model_id": "provider-alpha/model-advanced-v2",
  "selected_model_tier": "ADVANCED",
  "actual_input_tokens": 2041,
  "actual_output_tokens": 987,
  "actual_total_tokens": 3028,
  "estimated_cost_usd": 0.38,
  "actual_cost_usd": 0.41,
  "cost_computation_method": "provider_api_reported",
  "authorized_cost_ceiling_usd": 1.00,
  "ceiling_exceeded": false,
  "chain_id": "chain-pipeline-20260428-00041",
```

```
  "chain_step": 2
}
```

9. Governance and Authorization

9.1. Policy Authority Model

RMRP defines a two-role authorization model for routing policy governance:

Policy Authority (PA): The entity authorized to issue, sign, update, and revoke Routing Policy Documents within a defined scope. A Policy Authority MUST be identified by a stable "policy_authority_id" and MUST possess a cryptographic signing key pair.

Budget Authority (BA): The entity authorized to approve inference expenditure for one or more cost centers. A Budget Authority is referenced by "budget_authority_id" in RPDs and MRDs. The relationship between Budget Authorities and cost centers is defined externally to RMRP.

A single organizational entity MAY hold both Policy Authority and Budget Authority roles. Implementations MAY define additional roles using the "extensions" mechanism.

9.2. Policy Issuance and Signing

RPDs MUST be signed by the Policy Authority using a digital signature mechanism before they are made available to Routing Engines. This specification RECOMMENDS JWS [RFC7515] with algorithm RS256 or ES256.

Routing Engines MUST verify the RPD signature before applying any policy. Routing Engines MUST reject unsigned or invalidly signed RPDs and write an ALR with outcome "POLICY_ERROR".

The public key or certificate used to verify RPD signatures MUST be provisioned to Routing Engines through a mechanism outside the scope of this specification. Key management practices SHOULD follow [RFC8551] or applicable organizational PKI policy.

9.3. Policy Versioning

RPD versions MUST follow semantic versioning. The full version string MUST be recorded in every MRD, ALR, and CAR produced under that policy version. This enables precise reconstruction of the routing governance context for any historical event.

When a Policy Authority issues a new RPD version, the new version MUST specify an "effective_date" in the future to allow Routing Engines time to load and validate the updated policy before it takes effect. A transition period of not less than 15 minutes between publication and "effective_date" is RECOMMENDED.

Routing Engines MAY cache active RPDs. Cached policies MUST be revalidated against the Policy Authority’s signing key upon each cache refresh. Cache TTL is implementation-defined but MUST NOT exceed the RPD "expiration_date".

9.4. Override Mechanisms

RMRP does not define a general-purpose override mechanism that permits callers to bypass routing policy. All routing decisions MUST be governed by a valid, signed RPD.

If a deployment requires the ability for privileged callers to escalate routing decisions (e.g., an operations team requesting ADVANCED tier for a specific task), this capability MUST be implemented as an explicit RPD rule with appropriate conditions, not as an out-of-band bypass.

Emergency override conditions, if required by an organization, MUST be defined in a dedicated RPD with a named Policy Authority and a "FULL" audit level for all events processed under that policy. Emergency RPDs MUST have short expiration windows.

10. Transport Considerations

10.1. HTTP Transport

When RMRP is used in conjunction with HTTP-based inference APIs, the following conventions APPLY:

The MRD SHOULD be attached to outbound inference requests using a custom HTTP header:

```
RMRP-MRD: <base64url-encoded JSON MRD>
```

Where base64url encoding is as defined in [RFC4648]. Note: This header field name does not use the "X-" prefix, consistent with the guidance in [RFC6648] deprecating the "X-" convention for newly defined header fields.

If the MRD exceeds HTTP header size limits, it MAY be included as a JSON object in the request body under the reserved key "_rmrp_mrd", provided the inference API accepts JSON request bodies.

The "mrd_id" SHOULD be returned in the inference response using a custom HTTP header:

```
RMRP-MRD-ID: <mrd_id value>
```

HTTP responses from the Routing Engine to the caller SHOULD include the "mrd_id" and "request_id" for correlation.

RMRP error responses in HTTP transport SHOULD use the Problem Details format defined in [RFC9457] with the following fields:

```
type: A URI identifying the RMRP error class.
title: A human-readable RMRP error code (e.g., "RMRP-004").
status: The applicable HTTP status code.
detail: A human-readable error description.
instance: A URI reference to the specific routing event.
```

10.2. Header Propagation

In multi-hop deployments where inference requests pass through intermediate systems before reaching the Routing Engine, the following APPLY:

- The "X-RMRP-MRD" header MUST be propagated unchanged through intermediate systems.
- Intermediate systems MUST NOT modify or strip the "X-RMRP-MRD" header.
- If an intermediate system performs its own routing, it MUST produce a new MRD and chain it to the original using the

"chain_id" mechanism.

10.3. Non-HTTP Transports

RMRP metadata structures are transport-agnostic. For non-HTTP transports (e.g., gRPC, AMQP, Kafka):

- The MRD MUST be attached as a structured metadata object in the transport envelope.
- The transport-specific mechanism for attaching metadata is implementation-defined but MUST be documented by the implementation.
- All other normative requirements of this specification apply regardless of transport.

11. Security Considerations

11.1. Policy Integrity

RPDs define the governance of all inference expenditure and model selection in a deployment. Unauthorized modification of an RPD could result in unauthorized use of high-cost model tiers, bypass of cost controls, or suppression of audit records. Implementations MUST enforce RPD signature verification as specified in Section 9.2. RPDs MUST be stored and transmitted in a manner that prevents unauthorized modification.

11.2. MRD Tampering

A tampered MRD could be used to misattribute inference costs or falsify audit records. In deployments with high-assurance requirements, Routing Engines SHOULD produce a cryptographic signature over each MRD using the Policy Authority’s signing key or a dedicated Routing Engine signing key. Receiving systems SHOULD verify this signature.

MRDs MUST NOT contain inference request content, prompt text, or user-supplied data. MRDs are governance metadata only.

11.3. Audit Log Integrity

ALR and CAR records MUST be written to a system that prevents modification or deletion by the Routing Engine itself or by operators without separate authorization. Implementations SHOULD implement hash chaining over the ALR sequence as described in Section 7.2, using the "previous_alr_id" and "alr_hash" fields. Implementations MAY anchor ALR hash roots to external immutable systems (e.g., transparency logs, public blockchains) for enhanced tamper-evidence.

11.4. Denial of Service

A malicious or malfunctioning caller could submit high-volume requests designed to maximize ADVANCED tier routing and exhaust budget ceilings. Routing Engines SHOULD implement rate limiting per source system and per cost center. Rate limiting thresholds are outside the scope of this specification.

The Routing Engine itself is a critical component. Its unavailability prevents all inference processing. Deployments SHOULD implement redundant Routing Engine instances. Routing Engines SHOULD implement circuit breakers for Audit Store connectivity, with defined behavior for the case where audit

records cannot be written (see RMRP-007).

11.5. Credential Exposure

RMRP records MUST NOT contain AI provider API keys, secrets, tokens, or authentication credentials. Model Identifiers in RMRP records are opaque strings and MUST NOT embed credentials. Authentication with model providers is a separate concern handled outside the RMRP governance layer.

12. Privacy Considerations

RMRP governance records (MRDs, ALRs, CARs) are operational metadata about routing decisions. They do not, and MUST NOT, contain the content of inference requests or responses.

However, the "source_system", "cost_center", and "task_type" fields in RMRP records may be sufficient to infer information about organizational activities or individual user behavior in certain deployment contexts. Implementations SHOULD apply access controls to the Audit Store consistent with the sensitivity of the operational data it contains.

In deployments subject to data residency requirements, implementations MUST ensure that ALR and CAR records are stored in jurisdictions consistent with applicable regulations. RMRP does not specify geographic constraints on record storage.

The "request_id" field, if it can be linked to an individual user, may constitute personal data under applicable privacy regulations. Organizations MUST assess whether RMRP records are subject to data subject rights obligations under applicable law and implement appropriate controls.

13. IANA Considerations

This document requests the following registrations:

HTTP Header Field Registration:

- Header Field Name: RMRP-MRD
- Status: Provisional
- Reference: This document, Section 10.1
- Change Controller: IETF

- Header Field Name: RMRP-MRD-ID
- Status: Provisional
- Reference: This document, Section 10.1
- Change Controller: IETF

Media Type Registration:

- Type name: application
- Subtype name: rmrp+json
- Required parameters: none
- Optional parameters: version
- Encoding considerations: binary (UTF-8 encoded JSON)
- Security considerations: See Section 11
- Interoperability considerations: none
- Published specification: This document
- Applications: AI model routing governance
- Additional information: none
- Contact: See Author’s Address
- Intended usage: COMMON

Change controller: IETF

URN Namespace for RMRP Error Types:

This document requests registration of a URN sub-namespace under "urn:ietf:params" per the process defined in [RFC8141] for use as "type" values in RMRP error responses per Section 10.1:

urn:ietf:params:rmrp:error:

Requested initial error type URNs pending IANA assignment:

urn:ietf:params:rmrp:error:policy-not-found
urn:ietf:params:rmrp:error:validation-failure
urn:ietf:params:rmrp:error:budget-exceeded
urn:ietf:params:rmrp:error:model-unavailable
urn:ietf:params:rmrp:error:fallback-exhausted
urn:ietf:params:rmrp:error:policy-expired
urn:ietf:params:rmrp:error:audit-store-failure

Note to RFC Editor: This section is to be updated to reflect actual IANA registry assignments prior to publication as an RFC.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457,

DOI 10.17487/RFC9457, July 2023,
<https://www.rfc-editor.org/rfc/rfc9457>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/rfc/rfc9562>.

14.2. Informative References

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC6648] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the 'X-' Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, DOI 10.17487/RFC6648, June 2012, <https://www.rfc-editor.org/rfc/rfc6648>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <https://www.rfc-editor.org/rfc/rfc8141>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/rfc/rfc8551>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[semver] Preston-Werner, T., "Semantic Versioning 2.0.0", 2013, <https://semver.org/>.

[ROUTELLM] Ong, I., Almahairi, A., Wu, V., Chiang, W., Wu, T., Gonzalez, J., Kadous, M., and I. Stoica, "RouteLLM: Learning to Route LLMs with Preference Data", LMSYS Blog, July 2024, <https://lmsys.org/blog/2024-07-01-routellm/>.

[PLPES] Reilly, L. J., "Protocol Layer Prompt Engineering Specification (PLPES)", Internet-Draft draft-reilly-plpes-00, April 2026, <https://datatracker.ietf.org/doc/draft-reilly-plpes/>.

[REM] Reilly, L. J., "Reilly EternaMark (REM) Protocol: Dual-Layer Digital Permanence for Intellectual Property", Internet-Draft draft-reilly-rem-protocol-01, <https://datatracker.ietf.org/doc/draft-reilly-rem-protocol/>.

Acknowledgments

The author acknowledges the foundational research contributions of the LLM routing research community, whose work establishing cost-quality trade-off frameworks for model selection provided essential context for this protocol-layer specification. This document addresses the governance and standardization layer above that body of research.

Author’s Address

Lawrence J. Reilly Jr.
Independent
Email: lawrencejohnreilly@gmail.com