Network Working Group
Internet-Draft
Intended status: Informational
Expires: 25 September 2026

A. Nunez
ThriveTech Services LLC

March 2026

# Mahalaxmi Federation and Orchestration Protocol (MFOP) draft-nunez-mfop-federated-ai-orchestration-00

## Abstract

This document defines the Mahalaxmi Federation and Orchestration Protocol (MFOP), a protocol for coordinating parallel artificial intelligence (AI) agent execution across a distributed network of heterogeneous compute nodes. MFOP specifies node identity and registration, capability advertisement, compliance-zone-aware job routing using database-layer enforcement, semantic input partitioning, cryptographically signed billing receipts, configurable economic settlement, and a layered security model comprising AI safety policy validation and execution sandbox isolation. The protocol is designed to operate across three simultaneous deployment configurations: private enterprise meshes, managed cloud pools, and open community marketplaces. MFOP is agnostic to the underlying AI model provider.

## Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document should take place at the MFOP discussion list maintained at https://mahalaxmi.ai/mfop/discuss. The current draft and revision history are maintained at https://mahalaxmi.ai/mfop/draft.

## 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 2 September 2026.



Table of Contents

## 1. Introduction

The growth of large language model (LLM) deployments across enterprise environments has created a need for a coordination layer that can span heterogeneous compute infrastructure while satisfying compliance, billing, and safety requirements that vary by jurisdiction and industry.

MFOP addresses this need by defining a protocol for federated AI orchestration. A federation consists of one or more compute nodes, each of which may be operated by different entities under different compliance regimes. A submitter -- a user, application, or automated system -- presents a job to the federation. The federation routes the job to an appropriate node based on the job's compliance zone requirements, the node's capability advertisement, and the economic terms in effect.

Existing distributed AI systems suffer from four specific limitations that this protocol addresses. First, they require complete trust between all participating nodes, making them unsuitable for multi-tenant or community-contributed compute deployments. Second, they lack fine-grained regulatory compliance zone enforcement at the data layer. Third, they provide no standardized economic model for compensating independent node operators with tamper-evident proof of work. Fourth, they do not provide a configurable, operator-adjustable economic fee model modifiable at runtime without code deployment.

This specification defines the wire protocol, data formats, cryptographic mechanisms, and behavioral requirements for all components of a conforming MFOP federation. The full protocol specification is maintained at [MFOP-SPEC].

## 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.

Federation: A logical grouping of one or more MFOP-conformant compute nodes operating under a shared governance configuration.

Node: A compute resource registered with a federation that accepts, executes, and returns AI workloads. A node may be a single server, a cluster, or a cloud compute pool.

Submitter: An entity (user, application, or automated system) that presents AI workloads to the federation for execution.

Compliance Zone: A named policy context that constrains job routing, data handling, and output validation. Defined zones: public, enterprise (SOC2), hipaa, sox, fedramp.

Job: A discrete unit of AI workload submitted to the federation for execution. A job carries a payload, compliance zone assertion, and billing authorization.

Receipt: A cryptographically signed record of a completed job execution, including token counts, timestamps, node identity, and billing amounts.

Economic Settlement: The process by which accumulated billing receipts are converted into financial transfers between submitters, node operators, and the platform.

PAK Key (Platform API Key): A bearer credential issued by the platform that authorizes access to federation API endpoints.

## 3. Node Identity and Registration

Each node in a MFOP federation is identified by a stable, globally unique node identifier (node_id). The node_id is a 128-bit UUID (version 4) [RFC9562] assigned at registration time and persists across node restarts and software upgrades.

### 3.1. Registration Flow

A node initiates registration by sending a NodeRegistrationRequest to the federation's registration endpoint (POST /v1/federation/nodes/register). The request MUST include:

* node_id: a candidate UUID (the federation MAY override this)
* operator_id: the UUID of the registering operator account
* display_name: a human-readable name (max 64 characters)
* public_key: an Ed25519 public key in base64url encoding
* capability_advertisement: a CapabilityAdvertisement object (see Section 4)
* compliance_zones: the set of compliance zones the node is certified to handle
* endpoint_url: the HTTPS URL at which the node accepts job submissions

The federation returns a NodeRegistrationResponse containing the assigned node_id, a registration_token for subsequent authenticated calls, and the federation's current billing configuration.

### 3.2. Re-registration and Key Rotation

Nodes MUST re-register when their Ed25519 key pair is rotated. During key rotation, the node submits a re-registration request with both the old and new public keys, signed with the old private key. The federation verifies the old-key signature before accepting the new key. There is a 24-hour overlap window during which receipts signed with either key are accepted.

### 3.3. Node Health and Deregistration

Nodes MUST send a heartbeat to POST /v1/federation/nodes/{id}/heartbeat at least once every 60 seconds. A node that misses three consecutive heartbeat windows is marked INACTIVE and excluded from routing. Nodes may deregister voluntarily via DELETE /v1/federation/nodes/{id}.

## 4. Capability Advertisement

A node's capability advertisement declares the AI models available on the node, the hardware characteristics relevant to job routing, and the compliance certifications held by the node operator.

### 4.1. CapabilityAdvertisement Object

The CapabilityAdvertisement object includes the following fields:

* models: an array of ModelDescriptor objects (see Section 4.2)
* hardware_class: one of { cpu, gpu_consumer, gpu_datacenter, tpu }
* vram_gb: total GPU VRAM in gigabytes (0 for CPU nodes)
* max_context_tokens: the maximum context window the node can service
* max_concurrent_jobs: the maximum number of simultaneous jobs
* compliance_certifications: an array of certification identifiers (e.g., "soc2-type2", "hipaa-baa", "fedramp-moderate")
* nemo_rails_version: the NeMo Guardrails runtime version installed

### 4.2. ModelDescriptor

Each model available on a node is described by a ModelDescriptor:

* model_id: a canonical model identifier string
* model_family: one of { llama, mistral, gemma, falcon, phi, custom }
* parameter_count_b: approximate parameter count in billions
* quantization: one of { fp16, bf16, int8, int4, none }
* context_window_tokens: the maximum context window for this model
* supports_tool_use: boolean
* supports_vision: boolean

### 4.3. Capability Refresh

Nodes MUST update their capability advertisement via PUT /v1/federation/nodes/{id}/capabilities whenever their available models or hardware configuration changes. The federation propagates updated capability advertisements to the routing layer within 30 seconds.

## 5. Compliance-Zone-Aware Job Routing

MFOP routes each job to a node that satisfies the job's compliance zone requirements. Compliance zone satisfaction is a hard constraint: a job MUST NOT be routed to a node that is not certified for the job's compliance zone.

### 5.1. Compliance Zones

MFOP defines five compliance zones, ordered from least to most restrictive:

* public: No compliance requirements beyond the baseline NeMo safety rails.

* enterprise (SOC2): Requires SOC 2 Type II certification. Adds data residency detection, API credential exfiltration detection, and access logging enforcement.

* hipaa: Requires HIPAA BAA. Adds PHI pattern detection, PHI de-identification, and minimum-necessary output checks.

* sox: Requires SOX compliance controls. Adds financial PII isolation, price prediction blocking, and MNPI detection.

* fedramp: Requires FedRAMP authorization. Adds CUI handling, export control detection, and classification marking enforcement.

### 5.2. Routing Algorithm

When a job is received, the routing layer executes the following algorithm:

1. Filter: Identify all nodes with status ACTIVE that are certified for the job's compliance zone.

2. Filter: Remove nodes whose max_context_tokens is less than the job's estimated token count.

3. Filter: Remove nodes whose max_concurrent_jobs is currently exhausted.

4. Score: For each remaining node, compute a routing score: score = w_latency x latency_score + w_cost x cost_score + w_affinity x affinity_score. Default weights: w_latency = 0.4, w_cost = 0.4, w_affinity = 0.2.

5. Select: Route to the highest-scoring node. On tie, select uniformly at random.

If no node satisfies all filters, the job is queued with a configurable timeout (default: 120 seconds). If no node becomes available within the timeout, the federation returns HTTP 503 with a Retry-After header.

### 5.3. Affinity Rules

Submitters MAY specify affinity rules in their job submission:

* node_affinity: a list of preferred node_ids (soft preference)

* anti_affinity: a list of node_ids to exclude (hard constraint)

* geography: a preferred geographic region (ISO 3166-1 alpha-2 country code)

Affinity rules affect only the affinity_score component; compliance zone certification and capacity remain hard constraints.

## 6. Semantic Input Partitioning

For jobs whose input exceeds a single node’s max_context_tokens, MFOP provides a semantic partitioning mechanism that splits the input into coherent sub-jobs, routes each sub-job independently, and aggregates the results.

### 6.1. Partition Strategies

MFOP defines three partition strategies:

* sliding_window: Splits the input into overlapping windows of configurable size and overlap. Suitable for tasks where context continuity at boundaries is important.

* semantic_boundary: Splits at detected semantic boundaries (paragraph breaks, section headers, topic transitions). Produces more coherent sub-jobs at the cost of variable sub-job sizes.

* task_decomposition: Interprets the input as a structured task list and routes each task as an independent sub-job. Requires the input to conform to the MFOP TaskList schema.

### 6.2. Partition Request

A submitter requests partitioned execution by setting partition_strategy in the job submission. The federation's partition engine splits the input, assigns sub-job IDs (parent_job_id + sequence number), and routes each sub-job independently. Sub-jobs inherit the compliance zone and billing authorization of the parent job.

### 6.3. Aggregation

Once all sub-jobs complete, the federation's aggregation layer assembles the results in sequence-number order. For sliding_window partitions, the aggregator de-duplicates content in the overlap regions using a longest-common-subsequence merge. The assembled result is returned to the submitter as a single JobResult with an array of sub_job_receipts.

## 7. Cryptographically Signed Billing Receipts

Every completed job execution produces a BillingReceipt signed by the executing node. Signed receipts are the authoritative record for economic settlement and dispute resolution.

### 7.1. Receipt Structure

A BillingReceipt contains:

* receipt_id: a UUID (version 4) unique to this receipt
* job_id: the UUID of the completed job
* node_id: the UUID of the executing node
* submitter_id: the UUID of the submitter
* model_id: the model used for execution
* compliance_zone: the compliance zone under which the job executed
* input_tokens: the number of input tokens processed
* output_tokens: the number of output tokens generated

* wall_time_ms: total execution time in milliseconds
* completed_at: RFC 3339 [RFC3339] timestamp of job completion
* fee_schedule_id: the UUID of the BillingFeeConfig in effect
* input_token_cost_usd: computed input token cost in USD (6 decimal places)
* output_token_cost_usd: computed output token cost in USD (6 decimal places)
* platform_fee_usd: the platform's fee for this job
* node_earnings_usd: the node operator's earnings
* total_cost_usd: total cost to the submitter

### 7.2. Signature Scheme

Receipts are signed using Ed25519. The node signs the canonical JSON serialization of the receipt (keys sorted lexicographically, no whitespace) with its registered private key. The signature is base64url-encoded and included in the receipt as the signature field.

The federation verifies the receipt signature upon receipt using the node's registered public key. Receipts with invalid signatures MUST be rejected and MUST trigger a node integrity alert.

### 7.3. Receipt Storage and Retrieval

The federation stores all receipts for a minimum of 7 years to support compliance audit requirements. Submitters may retrieve their receipts via GET /v1/federation/receipts. Node operators may retrieve receipts via GET /v1/federation/nodes/{id}/receipts.

## 8. Configurable Economic Settlement

MFOP separates billing (the accumulation of signed receipts) from settlement (the financial transfer of funds). Settlement is configurable and may occur on different schedules for different participant types.

### 8.1. BillingFeeConfig

The platform administrator configures fee rates via a BillingFeeConfig object. Each BillingFeeConfig has a version identifier and an effective date. A new config may be created at any time; it takes effect at the start of the next billing period. BillingFeeConfig fields:

* input_token_rate_usd_per_1k: USD charged per 1,000 input tokens
* output_token_rate_usd_per_1k: USD charged per 1,000 output tokens
* platform_fee_pct: the platform's percentage of total token cost (0-100)
* node_revenue_share_pct: the node operator's percentage (0-100)
* settlement_period_days: how often settlement runs
* minimum_payout_usd: minimum accumulated earnings before a payout is issued

### 8.2. Submitter Billing

Submitters are billed on a postpay basis. At the end of each settlement period, the federation aggregates all receipts for the submitter and charges the payment method on file. The invoice includes an itemized list of job receipts, grouped by compliance zone and model.

### 8.3. Node Operator Settlement

Node operators are paid out at the end of each settlement period, provided their accumulated earnings exceed the minimum_payout_usd threshold. Operators who do not meet the threshold roll their earnings forward to the next period.

## 9. Security Model

MFOP implements a three-layer security model: transport security, AI safety policy validation, and execution sandbox isolation.

### 9.1. Transport Security

All MFOP API endpoints MUST be served over HTTPS using TLS 1.3 or higher [RFC8446]. Mutual TLS (mTLS) is RECOMMENDED for node-to-federation communication in private enterprise mesh deployments. API authentication uses PAK Keys transmitted as the X-Channel-API-Key HTTP header. PAK Keys are 256-bit random values encoded in base64url.

### 9.2. AI Safety Policy Validation

All job inputs and outputs MUST be validated against NeMo Guardrails policies before execution and before delivery to the submitter. The baseline policy set (required for all compliance zones) includes jailbreak detection and blocking, harmful content detection, PII leakage detection in outputs, and prompt injection detection. Nodes MUST run the NeMo Guardrails runtime version specified in their capability advertisement. Nodes running outdated versions MUST be flagged as DEGRADED and excluded from routing for compliance zones requiring features not present in the installed version.

### 9.3. Execution Sandbox Isolation

Each job MUST execute in an isolated sandbox. Nodes MUST implement sandbox isolation using one of the following mechanisms: gVisor (runsc), RECOMMENDED for cloud deployments; Firecracker microVMs, RECOMMENDED for bare-metal deployments; or WASM (Wasmtime), permitted for CPU-only inference workloads. Sandboxes MUST be destroyed and recreated between jobs. Job-specific state MUST NOT persist between jobs.

### 9.4. Audit Logging

All job routing decisions, receipt signatures, and settlement events MUST be written to an append-only audit log. The audit log is cryptographically chained using SHA-256 hashes. Only append operations are permitted.

## 10. Wire Protocol

MFOP uses JSON over HTTPS for all API communication. WebSocket connections are supported for streaming job output.

### 10.1. Request and Response Format

All request and response bodies are UTF-8 encoded JSON. Requests MUST include Content-Type: application/json. Successful responses use HTTP 200 or 201. Error responses use the standard error envelope:

```
{ "error": { "code": "", "message": "", "details": {} } }
```

Standard error codes: UNAUTHORIZED, FORBIDDEN, NOT_FOUND, VALIDATION_ERROR, QUOTA_EXCEEDED, NO_ELIGIBLE_NODE, COMPLIANCE_VIOLATION, INTERNAL_ERROR.

### 10.2. Streaming Output

Nodes that support streaming output expose a WebSocket endpoint at wss://{node_endpoint}/v1/jobs/{id}/stream. The node streams token output as JSON-framed delta messages:

```
{ "type": "delta", "text": "...", "token_count": N }
```

The stream is terminated with a completion message:

```
{ "type": "done", "receipt": { ... } }
```

### 10.3. Idempotency

Job submission requests SHOULD include an Idempotency-Key header (UUID). If a request with the same Idempotency-Key is received within 24 hours, the federation returns the original response without re-executing the job.

## 11. IANA Considerations

This document has no IANA actions.

## 12. Security Considerations

MFOP uses Ed25519 asymmetric signatures for billing receipt integrity. The security of this mechanism depends on the secrecy of each node’s private key. Implementations SHOULD store private keys in the operating system keychain or a hardware security module (HSM) and MUST NOT transmit private keys over any network channel.

Replay attacks against billing receipts are mitigated by the platform-issued nonce mechanism described in Section 7.2. Implementations MUST enforce nonce uniqueness and expiration.

Compliance zone enforcement is implemented at the database query layer using a generalized inverted index (GIN) on the node's compliance_zones array column, ensuring that nodes not certified for a compliance zone cannot appear in the eligible node set regardless of application-layer logic. Implementations MUST additionally verify that the compliance_zone field in a completed receipt matches the compliance_zone declared in the original job submission.

The execution sandbox isolation requirement (Section 9.3) protects submitter credentials and work content from node operators. Credentials are injected at cycle initialization using a platform-issued session key and are not stored on the node filesystem. Nodes MUST report their sandbox implementation version in every heartbeat.

All MFOP API endpoints MUST be served over TLS 1.3 or higher [RFC8446]. mTLS is REQUIRED in fedramp compliance zone deployments.

The composite AI safety policy attestation hash (SHA-256 of the concatenation of baseline, compliance zone, and domain policy hashes) provides cryptographic evidence of the safety policy configuration active at execution time. Policy hash deviations detected via heartbeat MUST result in immediate node suspension.

The append-only audit log (Section 9.4) provides a tamper-evident record of all federation activity via SHA-256 chaining. Audit log storage MUST implement appropriate access controls to protect confidentiality of log contents in regulated environments.

## 13. References

### 13.1. 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/info/rfc2119>.

[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/info/rfc8174>.

[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/info/rfc8446>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[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/info/rfc9562>.

## 13.2. Informative References

[RFC8179] Bradner, S. and J. Contreras, "Intellectual Property Rights in IETF Technology", BCP 79, RFC 8179, DOI 10.17487/RFC8179, May 2017, <https://www.rfc-editor.org/info/rfc8179>.

[MFOP-SPEC] Nunez, A., "Mahalaxmi Federation and Orchestration Protocol Specification Version 1.0 Draft", ThriveTech Services LLC, March 2026.

## Appendix A: REST API Reference

This appendix lists the MFOP REST API endpoints. All endpoints require an X-Channel-API-Key header unless noted otherwise. Base path: /v1/federation

| Method and Path | Name | Description |
|---|---|---|
| POST /v1/federation/nodes/register | Node registration | Register a new node with the federation. |
| PUT /v1/federation/nodes/{id}/capabilities | Capability update | Update a node’s capability advertisement. |
| POST /v1/federation/nodes/{id}/heartbeat | Node heartbeat | Signal that the node is alive and accepting jobs. |
| DELETE /v1/federation/nodes/{id} | Node deregistration | Voluntarily deregister a node. |
| POST /v1/federation/jobs | Job submission | Submit a job to the federation for execution. |
| GET /v1/federation/jobs/{id} | Job status | Retrieve the current status and result of a job. |

| | | |
|---|---|---|
| GET /v1/federation/jobs/{id}/receipt | Job receipt | Retrieve the signed billing receipt for a completed job. |
| GET /v1/federation/receipts | Submitter receipts | List all receipts for the authenticated submitter. |
| GET /v1/federation/nodes/{id}/receipts | Node receipts | List receipts for jobs executed by the node. |
| GET /v1/federation/nodes/{id}/earnings | Provider earnings | Current period tokens, estimated earnings, last payout. |
| GET /v1/federation/submitters/billing | Submitter billing summary | Current period cost and next billing date. |
| PATCH /v1/admin/federation/billing-config | Update fee model | Admin only. Creates new BillingFeeConfig row. |

Table 1

## Appendix B: Compliance Zone Policy Requirements

Each compliance zone requires specific NeMo Guardrails policy capabilities beyond the baseline.

| Zone | Required Rails Beyond Baseline |
|---|---|
| public | Baseline only. |
| enterprise (SOC2) | Data residency marker detection. API credential exfiltration detection. Access logging enforcement. |
| hipaa | PHI pattern detection (names, DOB, MRN, ICD-10, diagnoses, insurance IDs). PHI de-identification rail. Minimum necessary output check. |
| sox | Financial PII isolation. Price prediction blocking. MNPI detection. |
| fedramp | CUI handling. Export control detection (EAR/ ITAR). Classification marking enforcement. |

Table 2

## Appendix C: IPR Disclosure

The author has filed provisional patent applications with the United States Patent and Trademark Office (USPTO) covering certain mechanisms described in this specification. These applications, filed by ThriveTech Services LLC in March 2026, cover: compliance-zone routing enforcement using a GIN-indexed database array column; cryptographically signed work unit receipts and append-only billing ledger; configurable economic settlement with four fee modes; composite AI safety policy attestation hash; reputation-weighted federated node dispatch with rolling-window scoring; self-healing AI orchestration via signed patch registry and meta-instance correction cycles; manager-worker consensus orchestration with PTY-native spawning and shell environment capture; and escrow-gated AI token metering with actual-consumption billing.

In accordance with BCP 79 [RFC8179], the author discloses these pending patent applications. ThriveTech Services LLC is willing to grant licenses to implementors of this specification on reasonable and non-discriminatory (RAND) terms. Parties seeking licensing information should contact: ami.nunez@mahalaxmi.ai. A formal IPR disclosure will be filed at https://datatracker.ietf.org/ipr/ concurrent with or prior to submission of this Internet-Draft.

Acknowledgements

The author wishes to acknowledge the NVIDIA NeMo team for the NeMo Guardrails platform, which provides the foundational AI safety infrastructure referenced in this specification.

Author’s Address

Ami Hoepner Nunez
ThriveTech Services LLC
West Palm Beach, FL
United States of America
Email: ami.nunez@mahalaxmi.ai
URI: https://mahalaxmi.ai/mfop/draft