OpenAPI AI Agents Standard (OSSA) - Foundation

Intelligent Agent Orchestration: A Standards-Based Framework for Multi-Agent AI Systems

Thomas Scola
Bluefly.io
Portland, Maine, USA
thomas@bluefly.io

Abstract

The proliferation of specialized AI agents in enterprise environments necessitates standardized orchestration mechanisms to coordinate their activities effectively. This paper presents the OpenAPI AI Agents Standard (OSSA), a comprehensive framework for intelligent agent orchestration that addresses fundamental challenges in multi-agent system coordination. We propose a three-tier progressive compliance model (Core, Governed, Advanced) that enables organizations to adopt agent orchestration incrementally while maintaining interoperability across diverse AI frameworks including MCP, LangChain, CrewAI, and AutoGen. The framework introduces capability-based agent routing, dynamic task decomposition, and standardized handoff protocols. Experimental evaluation across 50 specialized agents executing 1,000 multi-agent workflows demonstrates 34% reduction in orchestration overhead, 26% improvement in coordination efficiency, and 21% increase in task completion rates compared to proprietary solutions. The proposed standard provides vendor-neutral protocols that enable seamless integration while supporting enterprise governance requirements including ISO 42001 and NIST AI RMF compliance.

1. Introduction

The evolution of artificial intelligence from monolithic models to specialized agent-based systems represents a fundamental architectural shift in computational systems. Organizations increasingly deploy multiple specialized AI agents to handle complex workflows, creating critical challenges in coordination, resource allocation, and context management. Current approaches suffer from vendor lock-in, incompatible protocols, and inefficient orchestration mechanisms that limit scalability and increase operational costs.

The OpenAPI AI Agents Standard (OSSA) addresses these challenges through a vendor-neutral, framework-agnostic approach to agent orchestration. Unlike proprietary solutions that create isolated ecosystems, OSSA establishes open protocols enabling interoperability across diverse AI frameworks while supporting enterprise governance requirements.

This research makes four primary contributions:

1. A formal specification for progressive compliance in agent orchestration systems
2. Capability-based routing algorithms for optimal agent selection

3. Standardized handoff protocols minimizing context loss
4. Integration bridges for existing AI frameworks

2. Background and Related Work

2.1 Current Agent Frameworks

Existing agent frameworks demonstrate various limitations:

LangChain provides extensive tool integration but lacks standardized orchestration protocols. Agent coordination requires custom implementations, leading to fragmented solutions across deployments.

CrewAI supports multi-agent workflows but operates within a single framework paradigm, limiting interoperability with external systems.

AutoGen (Microsoft) enables conversational agent patterns but provides limited support for complex orchestration scenarios requiring dynamic agent selection.

Model Context Protocol (MCP) by Anthropic standardizes tool interfaces but does not address multi-agent coordination or resource optimization.

2.2 Orchestration Challenges

Multi-agent systems face several orchestration challenges:

- **Protocol Incompatibility**: Agents from different frameworks cannot communicate effectively
- **Static Workflows**: Inability to adapt to changing task requirements dynamically
- **Context Fragmentation**: Loss of contextual information during agent handoffs
- **Resource Inefficiency**: Suboptimal agent selection and resource allocation

3. The OSSA Framework

3.1 Architecture Overview

The OpenAPI AI Agents Standard defines a three-tier progressive compliance model:

```
None
apiVersion: oaas/standard
kind: Agent
metadata:
  name: code-analyzer
  tier: governed
  domain: software-development
spec:
  capabilities:
    - code-analysis
    - security-scanning
    - performance-profiling
```

```
  orchestration:
    can-lead: true
    can-delegate: true
    specialization-level: expert
  compliance:
    iso42001: compliant
    nist-ai-rmf: compliant
```

Core Tier provides basic agent discovery and invocation:

- Agent registration and discovery
- Basic capability declaration
- Simple request-response patterns

Governed Tier adds enterprise controls:

- Audit logging and compliance tracking
- Resource constraints and budgets
- Quality gates and validation

Advanced Tier enables sophisticated orchestration:

- Dynamic workflow generation
- Multi-agent coordination
- Adaptive resource allocation

3.2 Capability-Based Routing

The framework implements intelligent agent selection through capability matching:

```python
class CapabilityRouter:
    def select_optimal_agent(self, task, available_agents):
        # Calculate capability scores
        scores = []
        for agent in available_agents:
            capability_match = self.calculate_capability_match(
                task.required_capabilities,
                agent.capabilities
            )
```

```
            specialization_score = self.evaluate_specialization(
                task.domain,
                agent.specialization_areas
            )

            availability_score = self.check_availability(
                agent.current_load,
                agent.max_capacity
            )

            composite_score = (
                capability_match * 0.4 +
                specialization_score * 0.4 +
                availability_score * 0.2
            )

            scores.append((agent, composite_score))

        # Return agent with highest score
        return max(scores, key=lambda x: x[1])[0]
```

3.3 Standardized Handoff Protocol

OSSA defines efficient handoff mechanisms minimizing context loss:

Python

```
class HandoffProtocol:
    def prepare_handoff(self, source_agent, target_agent, context):
        handoff_packet = {
            'task_id': context.task_id,
            'source': source_agent.id,
            'target': target_agent.id,
            'context': {
                'state': context.current_state,
                'history': context.get_relevant_history(),
                'constraints': context.constraints
```

```
            },
            'metadata': {
                'timestamp': datetime.now(),
                'protocol_version': 'standard'
            }
        }

        # Validate handoff compatibility
        if not self.validate_compatibility(source_agent, target_agent):
            raise HandoffException("Incompatible agent protocols")

        return self.compress_handoff(handoff_packet)
```

4. Implementation

4.1 Framework Integration

OSSA provides integration bridges for existing frameworks:

Python

```
# LangChain Integration
class LangChainBridge(OSSABridge):
    def wrap_agent(self, langchain_agent):
        return OSSAAgent(
            native_agent=langchain_agent,
            capabilities=self.extract_capabilities(langchain_agent),
            adapter=self.create_langchain_adapter()
        )

# CrewAI Integration
class CrewAIBridge(OSSABridge):
    def wrap_crew(self, crew):
        agents = []
        for crew_agent in crew.agents:
            agents.append(self.wrap_agent(crew_agent))
        return OSSAWorkflow(agents=agents)
```

4.2 Dynamic Task Decomposition

The framework enables intelligent task breakdown:

```python
class TaskDecomposer:
    def decompose_task(self, task, available_agents):
        # Analyze task complexity
        complexity_analysis = self.analyze_complexity(task)

        # Identify subtasks
        subtasks = self.identify_subtasks(task, complexity_analysis)

        # Map subtasks to agents
        task_assignments = []
        for subtask in subtasks:
            optimal_agent = self.capability_router.select_optimal_agent(
                subtask,
                available_agents
            )
            task_assignments.append({
                'subtask': subtask,
                'agent': optimal_agent,
                'priority': subtask.priority,
                'dependencies': subtask.dependencies
            })

        # Generate execution plan
        return self.generate_execution_plan(task_assignments)
```

5. Evaluation

5.1 Experimental Setup

We evaluated OSSA across three dimensions:

- **Orchestration Efficiency**: Overhead and coordination metrics
- **Task Performance**: Completion rates and quality scores
- **Interoperability**: Cross-framework communication success

Test Environment:

- 50 specialized agents across 5 frameworks
- 1,000 multi-agent workflows
- Tasks: Code generation, testing, documentation, analysis
- Baselines: Native framework orchestration, custom integrations

5.2 Results

Metric	Baseline	OSSA	Improvement
Orchestration Overhead	450ms	297ms	34% reduction
Coordination Efficiency	0.72	0.91	26% improvement
Task Completion Rate	78%	94%	21% increase
Context Preservation	65%	89%	37% improvement
Cross-Framework Success	45%	92%	104% improvement

5.3 Case Study: Multi-Framework Development Pipeline

Scenario: Coordinate agents from LangChain (planning), CrewAI (implementation), and AutoGen (testing) for feature development.

Baseline Approach: Custom integration scripts, manual handoffs

- Time: 45 minutes
- Success Rate: 65%
- Manual Interventions: 8

OSSA Approach: Standardized orchestration

- Time: 28 minutes (38% faster)
- Success Rate: 92%
- Manual Interventions: 1

6. Discussion

The evaluation demonstrates OSSA's effectiveness in addressing key orchestration challenges. The 34% reduction in overhead validates the efficiency of standardized protocols, while 104% improvement in cross-framework communication confirms the value of vendor-neutral standards.

Key findings:

1. **Progressive Compliance Enables Adoption**: Organizations can start with Core tier and advance gradually
2. **Capability Routing Improves Selection**: 26% better agent utilization through intelligent matching
3. **Standardized Handoffs Preserve Context**: 37% improvement in context retention

Limitations include initial integration overhead and the need for framework-specific adapters. Future work will address automatic adapter generation and machine learning-based optimization.

7. Conclusion

The OpenAPI AI Agents Standard provides a comprehensive framework for multi-agent orchestration, addressing critical challenges in coordination, interoperability, and resource optimization. Through progressive compliance tiers, capability-based routing, and standardized protocols, OSSA enables efficient orchestration while maintaining vendor neutrality. Experimental validation demonstrates significant improvements in orchestration efficiency, task performance, and cross-framework compatibility, establishing OSSA as a practical foundation for enterprise multi-agent systems.

Full spec

The framework enables intelligent task breakdown:

```python
openapi: 3.1.0
info:
  title: OSSA - Open Standards for Scalable Agents
  version: 0.1.8
  description: |
    Unified specification combining OSSA 360° Feedback Loop with Agent Capability Description Language (ACDL).
    This comprehensive standard enables interoperable, self-improving agent systems with governance,
    multi-protocol support, and token-efficient communication.

    Key Features:
    - 360° Feedback Loop (Plan → Execute → Review → Learn → Govern)
    - Agent Capability Description Language (ACDL) for interoperability
    - Multi-protocol support (REST, gRPC, WebSocket)
    - Token-efficient design with Props tokens and delta-first patterns
    - Enterprise governance with audit trails and budget management
    - DITA-native documentation with machine-lean roadmap integration

  contact:
    name: Bluefly.io
    email: thomas@bluefly.io
```

```yaml
    url: https://bluefly.io

  license:
    name: MIT
    url: https://opensource.org/licenses/MIT

  x-ossa-metadata:
    specification: OSSA
    acdl-version: "1.0.0"
    roadmap-version: "0.1.8"
    theme: "Foundations & Minimal Viable Standard"
    principles:
      - Interoperability-first (no framework rewrites required)
      - Token-efficiency by design (budgets, deltas, IDs over blobs)
      - Auditability and governance as first-class concepts
      - Docs-native (DITA) with machine-lean JSON sitemap
      - Portable agent taxonomy (roles, subtypes, capabilities)

servers:
  - url: https://api.ossa.bluefly.io/v1
    description: Production OSSA API
    x-protocols: [https]
  - url: grpc://grpc.ossa.bluefly.io:50051
    description: gRPC endpoint
    x-protocols: [grpc]
  - url: wss://ws.ossa.bluefly.io/realtime
    description: WebSocket real-time endpoint
    x-protocols: [websocket]
  - url: http://localhost:8080/v1
    description: Local development
    x-protocols: [http]

x-agent-capabilities:
  taxonomy:
```

```yaml
  version: "1.0.0"
  categories:
    orchestration:
      - orchestrator
      - router
      - scheduler
    execution:
      - worker
      - executor
      - processor
    evaluation:
      - critic
      - verifier
      - judge
    learning:
      - trainer
      - synthesizer
      - optimizer
    governance:
      - governor
      - auditor
      - enforcer
    telemetry:
      - monitor
      - collector
      - analyzer
    integration:
      - adapter
      - translator
      - bridge

capability-domains:
  nlp:
    - text-generation
```

```
      - summarization
      - translation
      - sentiment-analysis
      - entity-extraction
    vision:
      - object-detection
      - face-recognition
      - ocr
      - scene-understanding
    reasoning:
      - mathematical
      - logical
      - causal
      - temporal
    data:
      - retrieval
      - transformation
      - validation
      - persistence

x-uri-conventions:
  artifact: "artifact://{repo}/{path}@{commit}"
  vector: "vec://{space}/{id}"
  dita: "dita://{collection}/{topicId}"
  props: "@{namespace}:{project}:{version}:{id}"
  workspace: "workspace://{project}/.agents-workspace/{category}/{id}"

x-token-efficiency:
  strategies:
    - Key-based context (pass IDs not docs)
    - Delta-first prompts
    - Tiered prompting (shallow→deep)
    - Critic-on-outputs (lint/test) not full artifacts
    - Cacheable policy/style capsules (versioned)
```

```
    - Vector pre-filters (top-k IDs, expand late)
    - Pre-LLM validators (rules/regex/schema)
    - Payload compression allowed (zstd/base64)
    - Checkpoint memos instead of full history
    - Early-exit heuristics

  budget-defaults:
    maxTokensTask: 12000
    maxTokensSubtask: 4000
    maxTokensPlanning: 2000

x-grpc-service:
  proto: ossa.proto
  package: ossa.v1
  services:
    - name: AgentService
      methods:
        - Plan
        - Execute
        - Review
        - Learn
        - Govern
    - name: RegistryService
      methods:
        - Register
        - Discover
        - Update
        - Health

x-websocket-channels:
  /stream/execution:
    subscribe:
      message:
        $ref: '#/components/schemas/ExecutionStatus'
```

```
    publish:
      message:
        $ref: '#/components/schemas/ExecutionCommand'
    heartbeat:
      interval: 30000
      timeout: 5000

  /stream/feedback:
    subscribe:
      message:
        $ref: '#/components/schemas/FeedbackPacket'
    publish:
      message:
        $ref: '#/components/schemas/FeedbackRequest'

tags:
  - name: Registry
    description: Agent registration and discovery (ACDL)
  - name: Planning
    description: Task planning and strategy operations
  - name: Execution
    description: Task execution and monitoring
  - name: Feedback
    description: Multi-source critique and review
  - name: Learning
    description: Adaptation and improvement
  - name: Governance
    description: Budget, compliance, and audit
  - name: Memory
    description: Context and state management
  - name: Props
    description: Token resolution and expansion
  - name: Workspace
    description: .agents-workspace management
```

```yaml
paths:
  # Registry & Discovery (ACDL)
  /registry/agents:
    post:
      tags: [Registry]
      summary: Register agent with capabilities
      operationId: registerAgent
      x-agent-operation:
        capability: registry-management
        complexity: low
        cacheable: false
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/AgentRegistration'
      responses:
        '201':
          description: Agent registered
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/RegisteredAgent'

    get:
      tags: [Registry]
      summary: Discover agents by capability
      operationId: discoverAgents
      x-agent-operation:
        capability: registry-search
        complexity: low
        cacheable: true
```

```
      parameters:
        - name: capability
          in: query
          required: true
          schema:
            type: string
          description: Required capability
        - name: version
          in: query
          schema:
            type: string
          description: Version constraint (e.g., ">=2.0.0")
        - name: agentType
          in: query
          schema:
            type: string
            enum: [orchestrator, worker, critic, verifier, judge, integrator, trainer,
governor, telemetry]
        - name: maxLatency
          in: query
          schema:
            type: integer
          description: Maximum acceptable latency in ms
      responses:
        '200':
          description: Compatible agents found
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/AgentMatch'

  /registry/agents/{agentId}:
    get:
```

```yaml
    tags: [Registry]
    summary: Get agent specification
    operationId: getAgentSpec
    parameters:
      - name: agentId
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Agent specification
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/AgentSpecification'

  patch:
    tags: [Registry]
    summary: Update agent registration
    operationId: updateAgentRegistration
    parameters:
      - name: agentId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AgentUpdate'
    responses:
```

```
      '200':
        description: Registration updated

/registry/health/{agentId}:
  post:
    tags: [Registry]
    summary: Report agent health
    operationId: reportHealth
    parameters:
      - name: agentId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/HealthReport'
    responses:
      '204':
        description: Health recorded

# Planning
/plan:
  post:
    tags: [Planning]
    summary: Create execution plan
    operationId: createPlan
    x-agent-operation:
      capability: planning
      complexity: high
      requiresBudget: true
```

```yaml
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/PlanRequest'
    responses:
      '200':
        description: Execution plan created
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Plan'
      '402':
        $ref: '#/components/responses/InsufficientBudget'

/plan/{planId}/validate:
  post:
    tags: [Planning]
    summary: Validate plan feasibility
    operationId: validatePlan
    parameters:
      - name: planId
        in: path
        required: true
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ValidationRequest'
    responses:
```

```yaml
      '200':
        description: Validation results
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ValidationResult'

# Execution
/execute:
  post:
    tags: [Execution]
    summary: Execute plan or task
    operationId: execute
    x-agent-operation:
      capability: execution
      complexity: variable
      requiresBudget: true
      auditable: true
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ExecutionRequest'
    responses:
      '200':
        description: Execution started
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ExecutionReport'
      '402':
        $ref: '#/components/responses/InsufficientBudget'
```

```yaml
/execute/{executionId}/status:
  get:
    tags: [Execution]
    summary: Get execution status
    operationId: getExecutionStatus
    parameters:
      - name: executionId
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Current execution status
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ExecutionStatus'

# Feedback & Review
/feedback:
  post:
    tags: [Feedback]
    summary: Submit feedback
    operationId: submitFeedback
    x-agent-operation:
      capability: feedback-collection
      complexity: low
      auditable: true
    requestBody:
      required: true
      content:
        application/json:
          schema:
```

```
            $ref: '#/components/schemas/FeedbackPacket'
    responses:
      '201':
        description: Feedback recorded
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FeedbackResponse'

/review:
  post:
    tags: [Feedback]
    summary: Create execution review
    operationId: createReview
    x-agent-operation:
      capability: review-generation
      complexity: medium
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ReviewRequest'
    responses:
      '200':
        description: Review created
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Review'

/judge:
  post:
    tags: [Feedback]
```

```yaml
    summary: Judge execution quality
    operationId: judgeExecution
    x-agent-operation:
      capability: judgment
      complexity: high
      requiresEvidence: true
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/JudgmentRequest'
    responses:
      '200':
        description: Judgment rendered
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/JudgmentDecision'

# Learning
/learn:
  post:
    tags: [Learning]
    summary: Trigger learning cycle
    operationId: triggerLearning
    x-agent-operation:
      capability: learning
      complexity: high
      async: true
    requestBody:
      required: true
      content:
        application/json:
```

```yaml
          schema:
            $ref: '#/components/schemas/LearningRequest'
    responses:
      '202':
        description: Learning cycle initiated
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/LearningJob'

/learn/signals:
  post:
    tags: [Learning]
    summary: Submit learning signals
    operationId: submitLearningSignals
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              signals:
                type: array
                items:
                  $ref: '#/components/schemas/LearningSignal'
    responses:
      '201':
        description: Signals recorded

# Governance
/governance/budget:
  get:
    tags: [Governance]
```

```yaml
    summary: Get current budgets
    operationId: getBudgets
    parameters:
      - name: agentId
        in: query
        schema:
          type: string
      - name: projectId
        in: query
        schema:
          type: string
    responses:
      '200':
        description: Budget information
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/Budget'

  post:
    tags: [Governance]
    summary: Create budget
    operationId: createBudget
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/BudgetRequest'
    responses:
      '201':
        description: Budget created
```

```
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Budget'

  /governance/budget/enforce:
    post:
      tags: [Governance]
      summary: Enforce budget limits
      operationId: enforceBudget
      x-agent-operation:
        capability: budget-enforcement
        complexity: low
        critical: true
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/BudgetEnforcement'
      responses:
        '200':
          description: Enforcement result
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/EnforcementResult'

  # Audit
  /audit:
    post:
      tags: [Governance]
      summary: Log audit event
      operationId: logAudit
```

```yaml
    x-agent-operation:
      capability: audit-logging
      complexity: low
      immutable: true
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AuditEvent'
    responses:
      '201':
        description: Audit logged

/audit/emit:
  post:
    tags: [Governance]
    summary: Emit audit trail
    operationId: emitAuditTrail
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AuditEmitRequest'
    responses:
      '200':
        description: Audit trail emitted
        content:
          application/jsonl:
            schema:
              type: string
              description: JSONL format audit trail
```

```yaml
# Props Token Resolution
/props/resolve:
  post:
    tags: [Props]
    summary: Resolve Props tokens
    operationId: resolveProps
    x-agent-operation:
      capability: token-resolution
      complexity: low
      cacheable: true
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              tokens:
                type: array
                items:
                  type: string
                example: ["@RM:OSSA:0.1.8:E-018-STD", "@DITA:spec:agent-taxonomy"]
    responses:
      '200':
        description: Resolved URIs
        content:
          application/json:
            schema:
              type: object
              additionalProperties:
                type: string

# Workspace Management
/workspace/init:
```

```yaml
  post:
    tags: [Workspace]
    summary: Initialize .agents-workspace
    operationId: initWorkspace
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              projectId:
                type: string
              template:
                type: string
                enum: [minimal, standard, enterprise]
    responses:
      '201':
        description: Workspace initialized
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/WorkspaceInfo'

/workspace/{projectId}/roadmap:
  get:
    tags: [Workspace]
    summary: Get project roadmap
    operationId: getRoadmap
    parameters:
      - name: projectId
        in: path
        required: true
        schema:
```

```
            type: string
      responses:
        '200':
          description: Project roadmap
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Roadmap'


components:
  schemas:
    # ACDL Agent Registration
    AgentRegistration:
      type: object
      required: [agentId, name, type, capabilities, endpoints]
      properties:
        agentId:
          type: string
          pattern: '^[a-z0-9-]+$'
        name:
          type: string
        type:
          type: string
          enum: [orchestrator, worker, critic, verifier, judge, integrator, trainer,
governor, telemetry]
        subtype:
          type: string
          example: "worker.drupal"
        capabilities:
          type: array
          items:
            $ref: '#/components/schemas/Capability'
        endpoints:
          type: array
          items:
```

```yaml
        $ref: '#/components/schemas/Endpoint'
    constraints:
      $ref: '#/components/schemas/AgentConstraints'
    performance:
      $ref: '#/components/schemas/PerformanceMetrics'
    compatibility:
      $ref: '#/components/schemas/CompatibilitySpec'

AgentSpecification:
  allOf:
    - $ref: '#/components/schemas/AgentRegistration'
    - type: object
      properties:
        acdlVersion:
          type: string
        registeredAt:
          type: string
          format: date-time
        lastHealthCheck:
          type: string
          format: date-time
        status:
          type: string
          enum: [active, inactive, unhealthy, maintenance]

Capability:
  type: object
  required: [domain, name]
  properties:
    domain:
      type: string
      example: "nlp"
    name:
      type: string
```

```
      example: "text-generation"
    version:
      type: string
      example: "2.1.0"
    models:
      type: array
      items:
        type: object
        properties:
          name:
            type: string
          version:
            type: string
          precision:
            type: string

Endpoint:
  type: object
  required: [url, protocol]
  properties:
    url:
      type: string
    protocol:
      type: string
      enum: [https, http, grpc, websocket]
    healthCheck:
      type: string
    authentication:
      type: string
      enum: [none, bearer, apikey, oauth2, mtls]

AgentConstraints:
  type: object
  properties:
```

```
      maxConcurrency:
        type: integer
      maxPayloadSize:
        type: integer
      timeout:
        type: integer
      rateLimit:
        type: object
        properties:
          requests:
            type: integer
          window:
            type: integer

  CompatibilitySpec:
    type: object
    properties:
      requires:
        type: array
        items:
          type: object
          properties:
            agent:
              type: string
            version:
              type: string
      provides:
        type: array
        items:
          type: object
          properties:
            interface:
              type: string
            version:
```

```yaml
            type: string

    AgentMatch:
      type: object
      properties:
        agentId:
          type: string
        name:
          type: string
        score:
          type: number
          minimum: 0
          maximum: 1
        capabilities:
          type: array
          items:
            $ref: '#/components/schemas/Capability'
        endpoints:
          type: array
          items:
            $ref: '#/components/schemas/Endpoint'

    # Core OSSA Objects
    PlanRequest:
      type: object
      required: [goal, context]
      properties:
        goal:
          type: string
        context:
          type: object
        propsTokens:
          type: array
          items:
```

```yaml
        type: string
      description: Props tokens to expand for context
    constraints:
      type: array
      items:
        $ref: '#/components/schemas/Constraint'
    budget:
      $ref: '#/components/schemas/BudgetAllocation'
    agentType:
      type: string
    priority:
      type: string
      enum: [low, medium, high, critical]

Plan:
  type: object
  required: [id, goal, steps, estimatedCost]
  properties:
    id:
      type: string
    goal:
      type: string
    steps:
      type: array
      items:
        $ref: '#/components/schemas/PlanStep'
    estimatedCost:
      $ref: '#/components/schemas/Cost'
    confidence:
      type: number
    alternatives:
      type: array
      items:
        $ref: '#/components/schemas/AlternativePlan'
```

```yaml
    workspace:
      type: string
      description: Workspace URI for plan artifacts

PlanStep:
  type: object
  required: [id, action, agentType]
  properties:
    id:
      type: string
    action:
      type: string
    agentType:
      type: string
    requiredCapabilities:
      type: array
      items:
        type: string
    dependencies:
      type: array
      items:
        type: string
    estimatedTokens:
      type: integer
    checkpoint:
      type: boolean

ExecutionRequest:
  type: object
  required: [planId]
  properties:
    planId:
      type: string
    agentId:
```

```yaml
      type: string
    propsContext:
      type: array
      items:
        type: string
    parameters:
      type: object
    deltaOnly:
      type: boolean
      default: false
    compressionEnabled:
      type: boolean
      default: true

ExecutionReport:
  type: object
  required: [id, planId, status, startedAt]
  properties:
    id:
      type: string
    planId:
      type: string
    status:
      type: string
      enum: [pending, running, completed, failed, cancelled]
    startedAt:
      type: string
      format: date-time
    completedAt:
      type: string
      format: date-time
    actualCost:
      $ref: '#/components/schemas/Cost'
    outputs:
```

```yaml
      type: array
      items:
        $ref: '#/components/schemas/StepOutput'
    checkpoints:
      type: array
      items:
        $ref: '#/components/schemas/Checkpoint'
    workspace:
      type: string

FeedbackPacket:
  type: object
  required: [executionId, source, type, content]
  properties:
    executionId:
      type: string
    source:
      type: string
      enum: [human, agent, system, automated]
    sourceAgent:
      type: string
      description: Agent ID if source is agent
    type:
      type: string
      enum: [success, failure, quality, efficiency, improvement, security, accessibility]
    content:
      type: string
    metrics:
      type: object
      additionalProperties:
        type: number
    evidence:
      type: array
      items:
```

```yaml
        type: string
    severity:
      type: string
      enum: [low, medium, high, critical]

Review:
  type: object
  required: [executionId, overallScore, dimensions]
  properties:
    executionId:
      type: string
    overallScore:
      type: number
      minimum: 0
      maximum: 1
    dimensions:
      type: object
      properties:
        quality:
          type: number
        efficiency:
          type: number
        security:
          type: number
        accessibility:
          type: number
        compliance:
          type: number
    feedback:
      type: array
      items:
        $ref: '#/components/schemas/FeedbackPacket'
    recommendations:
      type: array
```

```
      items:
        type: string

JudgmentRequest:
  type: object
  required: [executionId, criteria]
  properties:
    executionId:
      type: string
    criteria:
      type: array
      items:
        type: string
    evidence:
      type: array
      items:
        type: string
    threshold:
      type: number

JudgmentDecision:
  type: object
  required: [decision, confidence, rationale]
  properties:
    decision:
      type: string
      enum: [approve, reject, escalate]
    confidence:
      type: number
    rationale:
      type: string
    evidence:
      type: array
      items:
```

```
        type: string
    dissenting:
      type: array
      items:
        type: string

LearningSignal:
  type: object
  required: [executionId, signalType, value]
  properties:
    executionId:
      type: string
    signalType:
      type: string
      enum: [reward, penalty, correction, preference]
    value:
      type: number
    context:
      type: object
    skillUpdates:
      type: array
      items:
        type: string
    vectorUpdates:
      type: array
      items:
        type: string

Budget:
  type: object
  required: [id, name, total, used, period]
  properties:
    id:
      type: string
```

```yaml
      name:
        type: string
      projectId:
        type: string
      agentId:
        type: string
      total:
        $ref: '#/components/schemas/Cost'
      used:
        $ref: '#/components/schemas/Cost'
      remaining:
        $ref: '#/components/schemas/Cost'
      period:
        type: string
        enum: [hourly, daily, weekly, monthly, per_execution]
      handoffPolicy:
        type: string
        enum: [block, queue, delegate, escalate]
      thresholds:
        type: array
        items:
          type: object
          properties:
            percent:
              type: number
            action:
              type: string

  Cost:
    type: object
    required: [tokens, dollars]
    properties:
      tokens:
        type: integer
```

```
        dollars:
          type: number
        breakdown:
          type: object
          properties:
            inputTokens:
              type: integer
            outputTokens:
              type: integer
            planningTokens:
              type: integer
            computeSeconds:
              type: number


    AuditEvent:
      type: object
      required: [eventType, actor, action, timestamp]
      properties:
        id:
          type: string
        eventType:
          type: string
          enum: [execution.reported, review.submitted, judge.decision, learning.persisted,
budget.threshold.hit, audit.appended]
        actor:
          type: string
        agentType:
          type: string
        action:
          type: string
        resource:
          type: string
        projectId:
          type: string
        outcome:
```

```
      type: string
    metadata:
      type: object
    hash:
      type: string
      description: Hash chain for immutability
    timestamp:
      type: string
      format: date-time

WorkspaceInfo:
  type: object
  properties:
    projectId:
      type: string
    path:
      type: string
    structure:
      type: object
      properties:
        plans:
          type: string
        executions:
          type: string
        feedback:
          type: string
        learning:
          type: string
        audit:
          type: string
        roadmap:
          type: string
    initialized:
      type: string
```

```
      format: date-time

  Roadmap:
    type: object
    properties:
      project:
        type: string
      version:
        type: string
      theme:
        type: string
      epics:
        type: array
        items:
          type: object
          properties:
            id:
              type: string
            title:
              type: string
            description:
              type: string
            deliverables:
              type: array
              items:
                type: string
            dependencies:
              type: array
              items:
                type: string
      sitemap:
        type: object
        additionalProperties:
          type: string
```

```
        description: DITA or vector URI

# Supporting schemas
ValidationRequest:
  type: object
  properties:
    checkBudget:
      type: boolean
      default: true
    checkCapabilities:
      type: boolean
      default: true
    checkDependencies:
      type: boolean
      default: true

ValidationResult:
  type: object
  required: [valid, issues]
  properties:
    valid:
      type: boolean
    issues:
      type: array
      items:
        type: object
        properties:
          type:
            type: string
          severity:
            type: string
          message:
            type: string
```

```yaml
BudgetAllocation:
  type: object
  properties:
    budgetId:
      type: string
    maxCost:
      $ref: '#/components/schemas/Cost'

BudgetEnforcement:
  type: object
  required: [budgetId, requestedCost]
  properties:
    budgetId:
      type: string
    requestedCost:
      $ref: '#/components/schemas/Cost'
    executionId:
      type: string
    overrideToken:
      type: string

EnforcementResult:
  type: object
  required: [approved, remaining]
  properties:
    approved:
      type: boolean
    remaining:
      $ref: '#/components/schemas/Cost'
    reason:
      type: string
    alternativeAgent:
      type: string
    routingDecision:
```

```yaml
      type: string

StepOutput:
  type: object
  properties:
    stepId:
      type: string
    status:
      type: string
    output:
      type: object
    tokenUsage:
      type: integer

Checkpoint:
  type: object
  properties:
    id:
      type: string
    stepId:
      type: string
    state:
      type: object
    memo:
      type: string
      description: Compressed summary instead of full history

Constraint:
  type: object
  properties:
    type:
      type: string
    value:
      type: string
```

```yaml
    priority:
      type: string

AlternativePlan:
  type: object
  properties:
    steps:
      type: array
      items:
        $ref: '#/components/schemas/PlanStep'
    estimatedCost:
      $ref: '#/components/schemas/Cost'
    tradeoffs:
      type: array
      items:
        type: string

ExecutionStatus:
  type: object
  properties:
    executionId:
      type: string
    status:
      type: string
    progress:
      type: number
    currentStep:
      type: string

FeedbackResponse:
  type: object
  properties:
    id:
      type: string
```

```
    receivedAt:
      type: string
      format: date-time

LearningRequest:
  type: object
  properties:
    executionIds:
      type: array
      items:
        type: string
    learningType:
      type: string
    targetMetrics:
      type: array
      items:
        type: string

LearningJob:
  type: object
  properties:
    id:
      type: string
    status:
      type: string
    createdAt:
      type: string
      format: date-time

BudgetRequest:
  type: object
  required: [name, total, period]
  properties:
    name:
```

```
      type: string
    total:
      $ref: '#/components/schemas/Cost'
    period:
      type: string

ReviewRequest:
  type: object
  required: [executionId]
  properties:
    executionId:
      type: string
    critics:
      type: array
      items:
        type: string

AuditEmitRequest:
  type: object
  properties:
    startTime:
      type: string
      format: date-time
    endTime:
      type: string
      format: date-time
    eventTypes:
      type: array
      items:
        type: string
    format:
      type: string
      enum: [jsonl, json, csv]
```

```yaml
PerformanceMetrics:
  type: object
  properties:
    throughput:
      type: number
    latencyP99:
      type: number
    availability:
      type: number
    errorRate:
      type: number

HealthReport:
  type: object
  properties:
    status:
      type: string
      enum: [healthy, degraded, unhealthy]
    metrics:
      $ref: '#/components/schemas/PerformanceMetrics'
    timestamp:
      type: string
      format: date-time

AgentUpdate:
  type: object
  properties:
    status:
      type: string
    capabilities:
      type: array
      items:
        $ref: '#/components/schemas/Capability'
    endpoints:
```

```yaml
      type: array
      items:
        $ref: '#/components/schemas/Endpoint'

RegisteredAgent:
  type: object
  properties:
    agentId:
      type: string
    registrationId:
      type: string
    registeredAt:
      type: string
      format: date-time

ExecutionCommand:
  type: object
  properties:
    command:
      type: string
      enum: [pause, resume, cancel, checkpoint]
    executionId:
      type: string

FeedbackRequest:
  type: object
  properties:
    executionId:
      type: string
    requestedFrom:
      type: array
      items:
        type: string
```

```yaml
responses:
  InsufficientBudget:
    description: Insufficient budget
    content:
      application/json:
        schema:
          type: object
          properties:
            error:
              type: string
            budgetId:
              type: string
            requested:
              $ref: '#/components/schemas/Cost'
            available:
              $ref: '#/components/schemas/Cost'
            suggestions:
              type: array
              items:
                type: string

securitySchemes:
  BearerAuth:
    type: http
    scheme: bearer
    bearerFormat: JWT

  ApiKey:
    type: apiKey
    in: header
    name: X-API-Key

  OAuth2:
    type: oauth2
```

```yaml
    flows:
      authorizationCode:
        authorizationUrl: https://auth.ossa.bluefly.io/authorize
        tokenUrl: https://auth.ossa.bluefly.io/token
        scopes:
          read: Read access
          write: Write access
          admin: Admin access

security:
  - BearerAuth: []
  - ApiKey: []
  - OAuth2: [read, write]

x-conformance:
  levels:
    bronze:
      requirements:
        - Basic OSSA object support
        - Plan/Execute/Review endpoints
        - JSON schema validation
    silver:
      requirements:
        - Full feedback loop
        - Budget enforcement
        - Audit logging
        - ACDL registration
    gold:
      requirements:
        - Multi-protocol support
        - Props token resolution
        - Learning signals
        - Workspace management
```

```
validation:
  endpoint: https://conformance.ossa.bluefly.io/validate
  badge: https://conformance.ossa.bluefly.io/badge/{level}/{agentId}
```