📚 API Documentation

Complete API documentation for the Agent Orchestration Operations system.

📋 Table of Contents

🌟 Overview

The Agent Orchestration Operations API provides programmatic access to manage and monitor agent operations, workflows, and system resources.

API Version

- **Current Version:** v1
- **Protocol:** REST over HTTPS
- **Data Format:** JSON
- **Authentication:** Bearer Token / API Key

🔐 Authentication

API Key Authentication

```
# Include API key in header
curl -H "Authorization: Bearer YOUR_API_KEY" \
     -H "Content-Type: application/json" \
     https://api.agent-orchestration-ops.com/v1/agents
```

OAuth 2.0 Authentication

```
# Get access token
curl -X POST https://api.agent-orchestration-ops.com/oauth/token \
     -H "Content-Type: application/json" \
     -d '{
       "grant_type": "client_credentials",
       "client_id": "your_client_id",
       "client_secret": "your_client_secret"
     }'

# Use access token
curl -H "Authorization: Bearer ACCESS_TOKEN" \
     https://api.agent-orchestration-ops.com/v1/agents
```

🌐 Base URL

```
Production: https://api.agent-orchestration-ops.com/v1
Staging: https://staging-api.agent-orchestration-ops.com/v1
Development: http://localhost:3000/v1
```

⚡ Rate Limiting

- **Default Limit:** 1000 requests per hour
- **Burst Limit:** 100 requests per minute
- **Headers:** Rate limit information in response headers

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 999
X-RateLimit-Reset: 1640995200
```

🚨 Error Handling

Error Response Format

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid request parameters",
    "details": [
      {
        "field": "email",
        "message": "Email is required"
      }
    ],
    "request_id": "req_123456789",
    "timestamp": "2024-01-01T00:00:00Z"
  }
}
```

HTTP Status Codes

Code	Description
200	Success
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
429	Too Many Requests
500	Internal Server Error

🤖 Endpoints

Agents Management

List Agents

```
GET /v1/agents
```

Parameters:
- `page` (integer): Page number (default: 1)
- `limit` (integer): Items per page (default: 20, max: 100)
- `status` (string): Filter by status (active, inactive, error)
- `type` (string): Filter by agent type

Response:

```json
{
  "data": [
    {
      "id": "agent_123",
      "name": "Data Processing Agent",
      "type": "data_processor",
      "status": "active",
      "created_at": "2024-01-01T00:00:00Z",
      "updated_at": "2024-01-01T12:00:00Z",
      "metrics": {
        "tasks_completed": 1250,
        "success_rate": 98.5,
        "avg_response_time": 150
      }
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 45,
    "pages": 3
  }
}
```

Get Agent Details

```
GET /v1/agents/{agent_id}
```

Response:

```json
{
  "data": {
    "id": "agent_123",
    "name": "Data Processing Agent",
    "type": "data_processor",
    "status": "active",
    "configuration": {
      "max_concurrent_tasks": 10,
      "timeout": 300,
      "retry_attempts": 3
    },
    "health": {
      "status": "healthy",
      "last_check": "2024-01-01T12:00:00Z",
      "uptime": 86400
    },
    "metrics": {
      "tasks_completed": 1250,
      "tasks_failed": 15,
      "success_rate": 98.5,
      "avg_response_time": 150,
      "memory_usage": 65.2,
      "cpu_usage": 23.1
    }
  }
}
```

Create Agent

```
POST /v1/agents
```

Request Body:

```json
{
  "name": "New Processing Agent",
  "type": "data_processor",
  "configuration": {
    "max_concurrent_tasks": 5,
    "timeout": 300,
    "retry_attempts": 3
  },
  "environment": "production"
}
```

Response:

```json
{
  "data": {
    "id": "agent_456",
    "name": "New Processing Agent",
    "type": "data_processor",
    "status": "initializing",
    "created_at": "2024-01-01T12:30:00Z"
  }
}
```

Update Agent

```
PUT /v1/agents/{agent_id}
```

Request Body:

```json
{
  "name": "Updated Agent Name",
  "configuration": {
    "max_concurrent_tasks": 15,
    "timeout": 600
  }
}
```

Delete Agent

```
DELETE /v1/agents/{agent_id}
```

Response:

```
{
  "message": "Agent deleted successfully",
  "deleted_at": "2024-01-01T13:00:00Z"
}
```

Tasks Management

List Tasks

```
GET /v1/tasks
```

Parameters:
- `agent_id` (string): Filter by agent ID
- `status` (string): Filter by status (pending, running, completed, failed)
- `priority` (string): Filter by priority (low, medium, high, critical)
- `created_after` (datetime): Filter tasks created after date
- `created_before` (datetime): Filter tasks created before date

Response:

```
{
  "data": [
    {
      "id": "task_789",
      "agent_id": "agent_123",
      "type": "data_processing",
      "status": "completed",
      "priority": "medium",
      "created_at": "2024-01-01T10:00:00Z",
      "started_at": "2024-01-01T10:01:00Z",
      "completed_at": "2024-01-01T10:05:00Z",
      "duration": 240,
      "result": {
        "records_processed": 1000,
        "success": true
      }
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 150,
    "pages": 8
  }
}
```

Create Task

```
POST /v1/tasks
```

Request Body:

```
{
  "agent_id": "agent_123",
  "type": "data_processing",
  "priority": "medium",
  "payload": {
    "source": "database",
    "query": "SELECT * FROM users WHERE active = true",
    "output_format": "json"
  },
  "schedule": {
    "type": "immediate"
  }
}
```

Get Task Status

```
GET /v1/tasks/{task_id}
```

Response:

```
{
  "data": {
    "id": "task_789",
    "agent_id": "agent_123",
    "type": "data_processing",
    "status": "running",
    "priority": "medium",
    "progress": 65,
    "created_at": "2024-01-01T10:00:00Z",
    "started_at": "2024-01-01T10:01:00Z",
    "estimated_completion": "2024-01-01T10:06:00Z",
    "logs": [
      {
        "timestamp": "2024-01-01T10:01:30Z",
        "level": "info",
        "message": "Processing batch 1 of 10"
      }
    ]
  }
}
```

Workflows Management

List Workflows

```
GET /v1/workflows
```

Response:

```
{
  "data": [
    {
      "id": "workflow_456",
      "name": "Data Pipeline Workflow",
      "description": "Complete data processing pipeline",
      "status": "active",
      "steps": [
        {
          "id": "step_1",
          "name": "Data Extraction",
          "agent_type": "data_extractor",
          "order": 1
        },
        {
          "id": "step_2",
          "name": "Data Processing",
          "agent_type": "data_processor",
          "order": 2
        }
      ],
      "created_at": "2024-01-01T00:00:00Z"
    }
  ]
}
```

Execute Workflow

```
POST /v1/workflows/{workflow_id}/execute
```

Request Body:

```
{
  "parameters": {
    "source_database": "production",
    "output_location": "s3://bucket/output/"
  },
  "priority": "high"
}
```

System Monitoring

System Health

```
GET /v1/health
```

Response:

```json
{
  "status": "healthy",
  "timestamp": "2024-01-01T12:00:00Z",
  "services": {
    "database": {
      "status": "healthy",
      "response_time": 15
    },
    "redis": {
      "status": "healthy",
      "response_time": 5
    },
    "message_queue": {
      "status": "healthy",
      "pending_messages": 25
    }
  },
  "metrics": {
    "active_agents": 12,
    "running_tasks": 8,
    "completed_tasks_today": 1250,
    "system_load": 0.65,
    "memory_usage": 72.3,
    "disk_usage": 45.1
  }
}
```

System Metrics

```
GET /v1/metrics
```

Parameters:
- `period` (string): Time period (1h, 24h, 7d, 30d)
- `metric` (string): Specific metric name

Response:

```json
{
  "data": {
    "period": "24h",
    "metrics": {
      "task_completion_rate": [
        {
          "timestamp": "2024-01-01T00:00:00Z",
          "value": 95.2
        }
      ],
      "response_time": [
        {
          "timestamp": "2024-01-01T00:00:00Z",
          "value": 150
        }
      ],
      "error_rate": [
        {
          "timestamp": "2024-01-01T00:00:00Z",
          "value": 1.5
        }
      ]
    }
  }
}
```

🔌 WebSocket API

Connection

```javascript
const ws = new WebSocket('wss://api.agent-orchestration-ops.com/v1/ws');

// Authentication
ws.onopen = function() {
  ws.send(JSON.stringify({
    type: 'auth',
    token: 'YOUR_API_KEY'
  }));
};
```

Real-time Events

```javascript
ws.onmessage = function(event) {
  const data = JSON.parse(event.data);

  switch(data.type) {
    case 'agent_status_change':
      console.log('Agent status changed:', data.payload);
      break;
    case 'task_completed':
      console.log('Task completed:', data.payload);
      break;
    case 'system_alert':
      console.log('System alert:', data.payload);
      break;
  }
};
```

Subscribe to Events

```
// Subscribe to specific agent events
ws.send(JSON.stringify({
  type: 'subscribe',
  channel: 'agent_events',
  agent_id: 'agent_123'
}));

// Subscribe to system-wide events
ws.send(JSON.stringify({
  type: 'subscribe',
  channel: 'system_events'
}));
```

🛠️ SDK Examples

Python SDK

```python
from agent_orchestration import Client

# Initialize client
client = Client(
    api_key='YOUR_API_KEY',
    base_url='https://api.agent-orchestration-ops.com/v1'
)

# List agents
agents = client.agents.list(status='active')
print(f"Found {len(agents)} active agents")

# Create task
task = client.tasks.create(
    agent_id='agent_123',
    type='data_processing',
    payload={'source': 'database'},
    priority='medium'
)

# Monitor task progress
while task.status in ['pending', 'running']:
    task = client.tasks.get(task.id)
    print(f"Task progress: {task.progress}%")
    time.sleep(5)

print(f"Task completed with result: {task.result}")
```

JavaScript SDK

```javascript
import { AgentOrchestrationClient } from '@agent-orchestration/sdk';

// Initialize client
const client = new AgentOrchestrationClient({
  apiKey: 'YOUR_API_KEY',
  baseUrl: 'https://api.agent-orchestration-ops.com/v1'
});

// List agents
const agents = await client.agents.list({ status: 'active' });
console.log(`Found ${agents.length} active agents`);

// Create and monitor task
const task = await client.tasks.create({
  agentId: 'agent_123',
  type: 'data_processing',
  payload: { source: 'database' },
  priority: 'medium'
});

// Real-time monitoring
client.tasks.watch(task.id, (updatedTask) => {
  console.log(`Task progress: ${updatedTask.progress}%`);

  if (updatedTask.status === 'completed') {
    console.log('Task completed:', updatedTask.result);
  }
});
```

cURL Examples

```bash
# List agents
curl -H "Authorization: Bearer YOUR_API_KEY" \
     "https://api.agent-orchestration-ops.com/v1/agents?status=active"

# Create task
curl -X POST \
     -H "Authorization: Bearer YOUR_API_KEY" \
     -H "Content-Type: application/json" \
     -d '{
       "agent_id": "agent_123",
       "type": "data_processing",
       "payload": {"source": "database"},
       "priority": "medium"
     }' \
     "https://api.agent-orchestration-ops.com/v1/tasks"

# Get task status
curl -H "Authorization: Bearer YOUR_API_KEY" \
     "https://api.agent-orchestration-ops.com/v1/tasks/task_789"
```

📊 Response Schemas

Agent Schema

```
{
  "type": "object",
  "properties": {
    "id": {"type": "string"},
    "name": {"type": "string"},
    "type": {"type": "string"},
    "status": {"type": "string", "enum": ["active", "inactive", "error"]},
    "configuration": {"type": "object"},
    "health": {
      "type": "object",
      "properties": {
        "status": {"type": "string"},
        "last_check": {"type": "string", "format": "date-time"},
        "uptime": {"type": "number"}
      }
    },
    "metrics": {
      "type": "object",
      "properties": {
        "tasks_completed": {"type": "number"},
        "success_rate": {"type": "number"},
        "avg_response_time": {"type": "number"}
      }
    }
  }
}
```

Task Schema

```
{
  "type": "object",
  "properties": {
    "id": {"type": "string"},
    "agent_id": {"type": "string"},
    "type": {"type": "string"},
    "status": {"type": "string", "enum": ["pending", "running", "completed", "failed"]
},
    "priority": {"type": "string", "enum": ["low", "medium", "high", "critical"]},
    "progress": {"type": "number", "minimum": 0, "maximum": 100},
    "payload": {"type": "object"},
    "result": {"type": "object"},
    "created_at": {"type": "string", "format": "date-time"},
    "started_at": {"type": "string", "format": "date-time"},
    "completed_at": {"type": "string", "format": "date-time"}
  }
}
```

🔒 Security

API Security Best Practices

1. **Always use HTTPS** in production
2. **Store API keys securely** - never in code

3. **Implement proper rate limiting** on client side
4. **Validate all inputs** before sending requests
5. **Handle errors gracefully** and don't expose sensitive information
6. **Use webhook signatures** to verify authenticity
7. **Implement proper logging** for audit trails

Authentication Security

```python
# Good: Store API key in environment variable
import os
api_key = os.getenv('AGENT_ORCHESTRATION_API_KEY')

# Bad: Hardcode API key
api_key = 'sk-1234567890abcdef'  # Never do this!
```

📞 Support

For API support and questions:

- **Documentation:** https://docs.agent-orchestration-ops.com (https://docs.agent-orchestration-ops.com)
- **Status Page:** https://status.agent-orchestration-ops.com (https://status.agent-orchestration-ops.com)
- **Support Email:** api-support@agent-orchestration-ops.com
- **GitHub Issues:** Repository Issues (https://github.com/Empire325Marketing/agent-orchestration-ops/issues)

API Documentation last updated: $(date)