API Reference Guide

Base URL: `http://localhost:8006/api/v1`

1. List All Agents

```
GET /agents
```

Purpose:
- Discover all available AI agents
- View agent capabilities and configurations
- Find appropriate agents for specific tasks

When to use:
- Application initialization
- Agent selection interfaces
- Capability discovery

Example use case:
Building a dashboard where users need to see all available AI capabilities and choose the appropriate agent for their task.

Request Example:
```curl
curl -X GET 'http://localhost:8006/api/v1/agents' \
  -H 'accept: application/json'
```

Response Example:
```json
{
  "agents": [
    {
      "id": "agent_google_search",
      "name": "Google Search Agent",
      "description": "AI agent for Google Search",
      "capabilities": ["web_search", "question_answering"],
      "tool_category": "Search",
      "icon_class": "fas fa-search",
      "icon_color": "#4285F4"
    }
  ]
}
```

2. Get Agent Details

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

Purpose:
- Get detailed information about a specific agent
- View agent configuration and capabilities
- Understand agent's specialization

When to use:
- Before agent execution
- Displaying agent details to users
- Configuring agent-specific interfaces

Example use case:
User has selected an agent and needs to see its full capabilities before proceeding with execution.

Request Example:
```curl
curl -X GET 'http://localhost:8006/api/v1/agents/agent_google_search' \
  -H 'accept: application/json'
```

Response Example:
```json
{
  "id": "agent_google_search",
  "name": "Google Search Agent",
  "description": "AI agent for Google Search",
  "tool_id": "google_search",
  "capabilities": ["web_search", "question_answering"],
  "default_model": "gpt-4o-latest",
  "prompt_template": "...",
  "config": {}
}
```

3. Execute Agent

```
POST /agents/execute
```

Purpose:
- Execute complex AI workflows
- Process multi-step tasks
- Handle tool-assisted operations

When to use:
- Complex search operations
- Multi-step analysis tasks
- Tasks requiring external tools
- Long-running operations

Example use case:
User wants to search the web, analyze results, and generate a summary with relevant images.

Request Example:
```json
{
  "agent_id": "agent_google_search",
  "input": {
    "query": "Latest developments in AI technology",
    "max_results": 5
  },
  "model": "gpt-4",
  "async_execution": true,
  "metadata": {
    "user_locale": "en-US",
    "priority": "normal"
  }
}
```

Response Example:
```json
{
  "execution_id": "c9093e05-ed7e-4edd-8d39-e88c65e1ce3d",
  "agent_id": "agent_google_search",
  "status": "completed",
  "result": {
    "response": {
      "text": "Here are the latest developments in AI technology...",
      "model": "gpt-4",
      "usage": {
        "total_tokens": 1107
      }
    },
    "workflow_steps": ["Search executed", "Results analyzed", "Summary generated"],
    "images": [
      {
        "url": "https://example.com/ai-image.jpg",
        "alt": "AI Technology Illustration"
      }
    ]
```

```
  }
}
```

4. Generate Prompt

```
POST /prompts/generate
```

Purpose:
- Create optimized prompts for AI models
- Ensure compliance and formatting
- Generate consistent prompt structures

When to use:
- Simple prompt creation
- Compliance-required content
- Standardized interactions
- Template-based generation

Example use case:
Creating a HIPAA-compliant prompt for generating medical report summaries.

Request Example:
```json
{
  "intent": "Generate a HIPAA-compliant medical report summary",
  "context": "Patient visit summary for internal records",
  "constraints": {
    "tone": "professional",
    "length": "medium",
    "format": "markdown"
  },
  "context_aware_params": {
    "domain": "healthcare",
    "audience": "medical professionals",
    "compliance": ["HIPAA"]
  }
}
```

Response Example:
```json
{
  "optimized_prompt": "As a medical professional summarizing patient
records...",
  "confidence_score": 0.95,
  "suggested_models": ["gpt-4", "claude-2"],
  "prompt_tokens_estimate": 150,
```

```
  "alternative_prompts": [
    "Alternative HIPAA-compliant prompt version..."
  ],
  "metadata": {
    "compliance_verified": true,
    "format_type": "medical_summary"
  }
}
```

5. Check Execution Status

```
GET /agents/execution/{execution_id}/status
```

Purpose:
- Monitor async agent executions
- Track operation progress
- Check completion status

When to use:
- During async operations
- Progress monitoring
- Completion checking

Example use case:
Polling for updates while an agent is processing a complex search request.

Request Example:
```curl
curl -X GET
'http://localhost:8006/api/v1/agents/execution/c9093e05-ed7e-4edd-8d39-e88c65e
1ce3d/status' \
  -H 'accept: application/json'
```

Response Example:
```json
{
  "execution_id": "c9093e05-ed7e-4edd-8d39-e88c65e1ce3d",
  "agent_id": "agent_google_search",
  "status": "running",
  "progress": 0.65,
  "message": "Processing search results",
  "start_time": "2025-05-30T07:05:06.975693",
  "end_time": null
}
```

6. Get Execution Logs

```
GET /agents/execution/{execution_id}/logs
```

Purpose:
- Debug agent executions
- Monitor detailed progress
- Audit agent operations

When to use:
- Troubleshooting
- Detailed monitoring
- Compliance auditing

Example use case:
Investigating why a particular agent execution produced unexpected results.

Request Example:
```curl
curl -X GET
'http://localhost:8006/api/v1/agents/execution/c9093e05-ed7e-4edd-8d39-e88c65e
1ce3d/logs' \
  -H 'accept: application/json'
```

Response Example:
```json
{
  "execution_id": "c9093e05-ed7e-4edd-8d39-e88c65e1ce3d",
  "agent_id": "agent_google_search",
  "logs": [
    {
      "timestamp": "2025-05-30T07:05:06.976083",
      "level": "info",
      "message": "Started execution of agent Google Search Agent",
      "details": null
    }
  ]
}
```

7. Get Execution Actions

```
GET /agents/execution/{execution_id}/actions
```

Purpose:
- View detailed agent actions

- Track decision-making process
- Audit agent behavior

When to use:
- Detailed auditing
- Process verification
- Understanding agent decisions

Example use case:
Generating an audit trail of all actions taken by an agent during execution.

Request Example:
```curl
curl -X GET
'http://localhost:8006/api/v1/agents/execution/c9093e05-ed7e-4edd-8d39-e88c65e
1ce3d/actions' \
  -H 'accept: application/json'
```

Response Example:
```json
{
  "execution_id": "c9093e05-ed7e-4edd-8d39-e88c65e1ce3d",
  "agent_id": "agent_google_search",
  "actions": [
    {
      "timestamp": "2025-05-30T07:05:06.976236",
      "action_type": "web_search",
      "description": "Executing Google search",
      "input_data": {
        "query": "Latest developments in AI technology"
      },
      "status": "completed",
      "duration_ms": 1500
    }
  ]
}
```

NOTE: Here's main difference between the 2 endpoints

Agent Execute (`/agents/execute`)
- **Use for**: Complex tasks requiring multiple steps, tool usage, and orchestration
- **Examples**: Web searches, image generation, data analysis
- **Features**:
  - Handles multiple steps automatically
  - Can use external tools and APIs
  - Provides detailed logs and actions

  - Supports async execution for long-running tasks

Prompt Generate (`/prompts/generate`)
- **Use for**: Creating optimized prompts for direct use with AI models
- **Examples**: Customer support templates, content generation guidelines
- **Features**:
  - Focuses on prompt engineering
  - Handles compliance requirements
  - Optimizes prompt structure
  - Provides alternative variations