OpenAPI to MCP Tools

Transform OpenAPI APIs into MCP Tools

Introduction

Amazon Bedrock AgentCore Gateway transforms existing REST APIs defined in OpenAPI specifications into Model Context Protocol (MCP) tools. This transformation allows AI agents to interact with external services through a unified, standardized interface without requiring custom integration code.

The Gateway acts as a translation layer that converts OpenAPI specifications into MCP-compatible tools, handling authentication, request routing, and protocol conversion automatically.

Why is this Important?

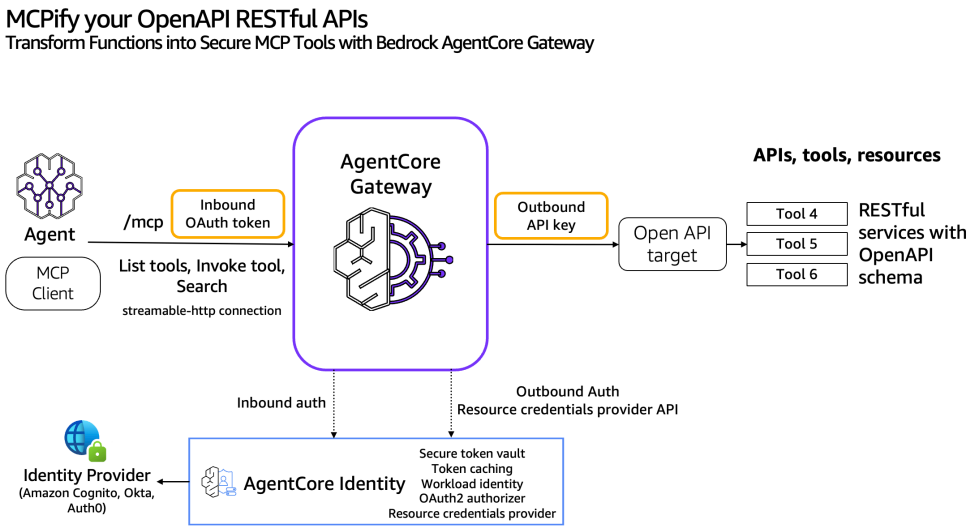
Modern enterprises rely on hundreds of REST APIs for their daily operations, but integrating these APIs with AI agents traditionally requires custom integration code, manual protocol conversion, and complex authentication management. AgentCore Gateway fundamentally transforms this by automating API integration through OpenAPI specification parsing, standardizing access via the MCP protocol, and centralizing authentication management.

In this lab, you will:

- Transform OpenAPI specifications into MCP-compatible tools using AgentCore Gateway
- Configure API key authentication for external service access
- Create secure gateway endpoints with inbound authentication
- Build a Mars Weather agent that leverages NASA's Open APIs
- Implement end-to-end agent workflows with external API integration

By the end of this lab, you should have a fully functional agent that can query Mars weather data through NASA APIs via your AgentCore Gateway, demonstrating real-world integration patterns that can be applied to any OpenAPI-compliant service.

High-Level Architecture



The architecture demonstrates a complete OpenAPI-to-MCP transformation workflow. The **Agent Application** uses Strands agents with Amazon Bedrock models for natural language processing. The **AgentCore Gateway** serves as the central transformation engine, converting OpenAPI specifications into MCP tools while managing authentication and routing. **Amazon Cognito** provides enterprise-grade inbound authentication, while **NASA APIs** represent external REST services requiring API key authentication.

Prerequisites & Setup

Environment Requirements

Before starting this lab, ensure you have:

- **Python 3.10+** installed on your system
- **Jupyter Notebook** environment for interactive development
- **AWS credentials** configured with appropriate permissions
- **Amazon Bedrock model access** (Anthropic Claude or Amazon Nova)
- **UV package manager** for Python dependency management

API Keys & Configuration Setup

NASA API Key Setup

To access NASA's Insight Weather API, you need a free API key:

1. **Visit NASA API Portal**: Navigate to https://api.nasa.gov/
2. **Request API Key**: Click "Get Started" and fill out the registration form
3. **Save Your Key**: Copy the generated API key for use in the lab
4. **Set Environment Variable**:

```
export NASA_API_KEY=your_nasa_api_key_here
```

NASA API Endpoints Used

- **InSight Weather API**: Provides Mars weather data from the InSight lander
- **Endpoint Format**: `https://api.nasa.gov/insight_weather/?api_key={API_KEY}&feedtype=json&ver=1.0`

AWS Configuration

Ensure your AWS credentials provide access to:

- **Amazon Bedrock**: For LLM model access
- **Amazon Cognito**: For identity provider creation
- **AgentCore Gateway**: For gateway creation and management

```
# Set AWS credentials (if not using IAM roles)
export AWS_ACCESS_KEY_ID=your_access_key
export AWS_SECRET_ACCESS_KEY=your_secret_key
export AWS_DEFAULT_REGION=us-west-2
```

Implementation

Core Gateway Concepts

Amazon Bedrock AgentCore Gateway provides a standardized way for AI agents to discover and interact with tools. Key concepts include:

Gateway Target - Defines the APIs or Lambda function that a Gateway will provide as tools to an agent.

OpenAPI Integration Features:

- Automatic parsing of OpenAPI 3.0+ specifications in JSON and YAML formats
- Seamless operation mapping to MCP tool definitions
- Comprehensive parameter validation based on OpenAPI schemas
- Intelligent response transformation to agent-friendly formats
- Sophisticated credential injection supporting API key, OAuth, and custom authentication methods

For complete implementation details, refer to the NASA Mars Weather notebook.

OpenAPI Specification Transformation

AgentCore Gateway automatically transforms OpenAPI specifications into MCP tools through intelligent parsing and mapping. The `operationId` field becomes the MCP tool name, while `parameters` define the input interface and `responses` specify the expected data structure.

The transformation process handles complex authentication schemes, parameter validation, and response formatting automatically, eliminating the need for developers to write custom integration code.

Gateway Creation and Configuration

Gateway creation establishes the foundation for secure API transformation. The `BedrockAgentCoreGatewayClient` provides the primary interface for gateway management, enabling developers to configure authentication, routing, and tool discovery.

The `customJWTAuthorizer` configuration enables OAuth-based authentication using Amazon Cognito or other OIDC-compliant identity providers, ensuring that only authenticated agents can access the gateway's tools.

Target Configuration with API Key Authentication

Target configuration connects the gateway to external APIs through flexible authentication schemes. The `openAPITargetConfiguration` accepts OpenAPI specifications and automatically configures routing, parameter validation, and response handling.

For API key authentication, developers specify the key location (query parameter, header, or cookie), the parameter name, and the actual key value. The gateway handles automatic injection of credentials for each request.

Different Approaches/Options

Authentication Method Variations

API Key Authentication (Current Example)

```
"authConfiguration": {
    "apiKeyAuth": {
        "apiKeyLocation": "query",  # or "header"
        "apiKeyName": "api_key",
        "apiKeyValue": nasa_api_key
    }
}
```

OAuth 2.0 Authentication

```
"authConfiguration": {
    "oauth2Auth": {
        "clientId": "your_oauth_client_id",
        "clientSecret": "your_oauth_secret",
        "tokenEndpoint": "https://api.service.com/oauth/token",
        "scope": "read:weather"
    }
}
```

Basic Authentication

```
"authConfiguration": {
    "basicAuth": {
        "username": "api_username",
        "password": "api_password"
    }
}
```

Framework Integration Options

Strands Agents | LangGraph | CrewAI

```
from strands import Agent, tool
from strands.models import BedrockModel

@tool
def weather_tool():
    return call_gateway_tool("getInsightWeather", {})

agent = Agent(model=BedrockModel("claude-sonnet-4"), tools=[weather_tool])
```

Try It Out

At an AWS Event

If you are following the workshop via workshop studio, now go to JupyterLab in SageMaker Studio. In the JupyterLab UI navigate to `02-AgentCore-gateway/02-transform-apis-into-mcp-tools/01-transform-openapi-into-mcp-tools/01-openapis-into-mcp-api-key.ipynb`

Self-paced

Congratulations!

Excellent work! You've successfully transformed a REST API into MCP tools using AgentCore Gateway and built a functional Mars Weather agent. Your implementation demonstrates how to bridge the gap between traditional REST APIs and modern AI agents through standardized protocols, creating a scalable foundation for enterprise API integration!

Previous Next

amit sharma

Diving Deep into Bedrock AgentCore

Event ends in 17 hours 50 minutes.