Tutorial 03: OpenAPI Tools - REST API Integration

Difficulty: beginner

Reading Time: 45 minutes

Tags: beginner, openapi, rest-api, tools, integration

Description: Integrate REST APIs into your agents using OpenAPI specifications for seamless external service integration. Connect to weather APIs, databases, and third-party services.

Tutorial 03: OpenAPI Tools - Connect Your Agent to Web APIs

Overview

Connect your agents to the entire web! Learn how to automatically generate tools from OpenAPI specifications, enabling your agent to interact with REST APIs without writing tool functions manually.

What You'll Build: A Chuck Norris fact assistant that:

- Searches for Chuck Norris jokes by category
- Gets random jokes
- Lists available categories
- Uses **OpenAPIToolset** to auto-generate tools from API specification

Why It Matters: Instead of manually writing tool functions for every API endpoint, OpenAPI specifications let ADK auto-generate tools, saving time and reducing errors.

Prerequisites

- Python 3.9+
- `google-adk` installed
- Google API key
- Completed Tutorials 01-02 (basics)
- Basic understanding of REST APIs

Core Concepts

What is OpenAPI?

OpenAPI (formerly Swagger) is a specification format for describing REST APIs:

```
{
  "openapi": "3.0.0",
  "paths": {
    "/jokes/random": {
      "get": {
        "summary": "Get random joke",
        "parameters": [...]
      }
    }
  }
}
```

How OpenAPIToolset Works

```
OpenAPI Spec → ADK Auto-Generation → Tools Available to Agent
```

Example:

```
toolset = OpenAPIToolset(spec=api_spec)
# ADK automatically creates:
# - get_jokes_random()
# - get_jokes_search()
# - get_jokes_categories()
```

Benefits:

- ✅ No manual tool writing
- ✅ Always matches API specification
- ✅ Handles authentication automatically
- ✅ Validates parameters
- ✅ Works with any OpenAPI-compliant API

Use Case: Chuck Norris Fact Assistant

Scenario: Build an agent that retrieves Chuck Norris jokes/facts from the public Chuck Norris API.

Why This API?:

- ✅ Free, no API key required
- ✅ Simple OpenAPI specification
- ✅ Great for learning
- ✅ Fun and engaging

API: https://api.chucknorris.io/

Implementation: tutorial_implementation/tutorial03 (https://github.com/raphaelmansuy/adk_training/tree/main/tutorial_implementation/tutorial03/) - Complete working example with tests

---

Implementation

Project Structure

```
chuck_norris_agent/
├── __init__.py
├── agent.py
├── .env
└── README.md
```

Complete Code

chuck_norris_agent/init**.py**:

```python
from .agent import root_agent

__all__ = ['root_agent']
```

chuck_norris_agent/agent.py:

```
"""
Chuck Norris Fact Assistant - OpenAPI Tools Demonstration

This agent demonstrates how to use OpenAPIToolset to automatically
generate tools from an API specification without writing tool functions.
"""

from google.adk.agents import Agent
from google.adk.tools.openapi_tool import OpenAPIToolset

# ============================================================================
# OPENAPI SPECIFICATION
# ============================================================================

# Chuck Norris API OpenAPI Specification
# Based on: https://api.chucknorris.io/
CHUCK_NORRIS_SPEC = {
    "openapi": "3.0.0",
    "info": {
        "title": "Chuck Norris API",
        "description": "Free JSON API for hand curated Chuck Norris facts",
        "version": "1.0.0"
    },
    "servers": [
        {
            "url": "https://api.chucknorris.io/jokes"
        }
    ],
    "paths": {
        "/random": {
            "get": {
                "operationId": "get_random_joke",
                "summary": "Get a random Chuck Norris joke",
                "description": "Retrieve a random joke from the database. Can
                "parameters": [
                    {
                        "name": "category",
                        "in": "query",
                        "description": "Filter jokes by category (optional)",
                        "required": False,
                        "schema": {
                            "type": "string"
                        }
                    }
                ],
                "responses": {
                    "200": {
                        "description": "Successful response",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "type": "object",
                                    "properties": {
                                        "icon_url": {"type": "string"},
                                        "id": {"type": "string"},
                                        "url": {"type": "string"},
                                        "value": {"type": "string"}
                                    }
                                }
                            }
                        }
                    }
                }
            }
        },
        "/search": {
            "get": {
                "operationId": "search_jokes",
                "summary": "Search for jokes",
                "description": "Free text search for jokes containing the quer
                "parameters": [
                    {
                        "name": "query",
                        "in": "query",
                        "description": "Search query (3+ characters required)"
                        "required": True,
                        "schema": {
                            "type": "string",
                            "minLength": 3
                        }
                    }
                ],
                "responses": {
                    "200": {
                        "description": "Successful response",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "type": "object",
                                    "properties": {
                                        "total": {"type": "integer"},
                                        "result": {
                                            "type": "array",
                                    "items": {
                                        "type": "object",
                                        "properties": {
                                            "icon_url": {"type": "stri
                                            "id": {"type": "string"},
                                            "url": {"type": "string"},
                                            "value": {"type": "string"
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    },
    "/categories": {
        "get": {
            "operationId": "get_categories",
            "summary": "Get all joke categories",
            "description": "Retrieve list of available joke categories.",
            "responses": {
                "200": {
                    "description": "Successful response",
                    "content": {
                        "application/json": {
                            "schema": {
                                "type": "array",
                                "items": {
                                    "type": "string"
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

# ==============================================================================
# OPENAPI TOOLSET
# ==============================================================================
# Create OpenAPIToolset from specification
# ADK will automatically generate 3 tools:
# - get_random_joke(category: Optional[str])
# - search_jokes(query: str)
# - get_categories()
chuck_norris_toolset = OpenAPIToolset(spec_dict=CHUCK_NORRIS_SPEC)

# =============================================================================
# AGENT DEFINITION
# =============================================================================

root_agent = Agent(
    name="chuck_norris_agent",
    model="gemini-2.0-flash",

    description="""
    Chuck Norris fact assistant that can retrieve jokes/facts from the
    Chuck Norris API using OpenAPI tools.
    """,

    instruction="""
    You are a fun Chuck Norris fact assistant!

    CAPABILITIES:
    - Get random Chuck Norris jokes (optionally filtered by category)
    - Search for jokes containing specific keywords
    - List all available joke categories

    STYLE:
    - Be enthusiastic and playful
    - Chuck Norris jokes are exaggerated for comedic effect
    - Format jokes clearly for easy reading
    - If search returns multiple results, show a few best ones

    WORKFLOW:
    - For random requests → use get_random_joke
    - For specific topics → use search_jokes with query
    - To see categories → use get_categories
    - For category-specific random → use get_random_joke with category paramet

    IMPORTANT:
    - Always extract the 'value' field from API response (that's the actual jo
    - If search finds 0 results, suggest trying a different keyword
    - Categories are lowercase (e.g., "dev", "movie", "food")
    """,

    # Pass the toolset to the agent
    tools=[chuck_norris_toolset]
)
```

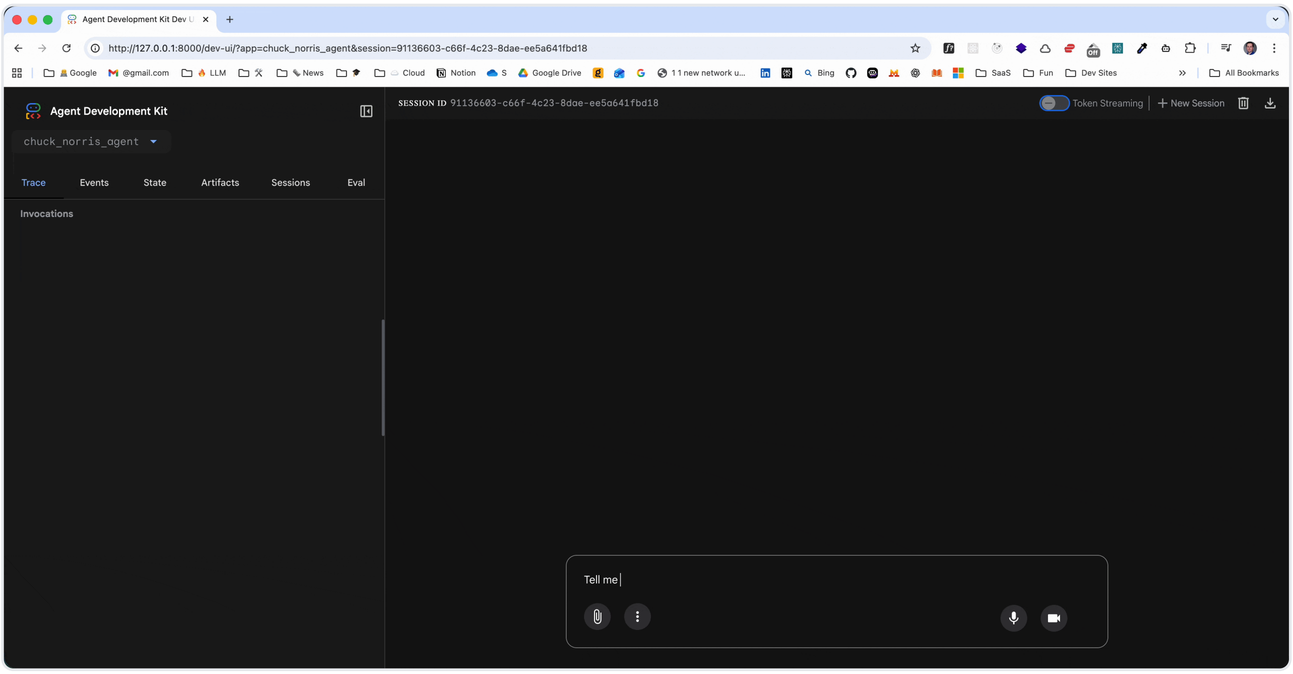
chuck_norris_agent/.env:

```
GOOGLE_GENAI_USE_VERTEXAI=FALSE
GOOGLE_API_KEY=your_api_key_here
```

Running the Agent

Demo in Action

Here's what your Chuck Norris agent looks like in action:


Agent Development Kit Dev UI
http://127.0.0.1:8000/dev-ui/?app=chuck_norris_agent&session=91136603-c66f-4c23-8dae-ee5a641fbd18
Agent Development Kit
chuck_norris_agent
Trace
Events
State
Artifacts
Sessions
Eval
Invocations
SESSION ID 91136603-c66f-4c23-8dae-ee5a641fbd18
Token Streaming
New Session
Tell me

Method 1: Web UI (Recommended)

```
cd /path/to/chuck_norris_agent/..
adk web chuck_norris_agent
```

Try these prompts:

- "Tell me a random Chuck Norris joke"
- "Find jokes about computers"
- "What joke categories exist?"
- "Give me a random movie joke"
- "Search for jokes with the word 'code'"

Method 2: Terminal

```
adk run chuck_norris_agent
```

Expected Behavior

Example 1: Random Joke

User: "Tell me a random Chuck Norris joke"

Agent Process:

1. Calls `get_random_joke()` tool (no category)
2. API returns JSON with 'value' field
3. Agent extracts and formats joke

Agent Response:

```
Here's a Chuck Norris fact for you:

"Chuck Norris can divide by zero."

😄 Classic!
```

Example 2: Search by Keyword

User: "Find jokes about programming"

Agent Process:

1. Calls `search_jokes(query="programming")` tool
2. API returns array of matching jokes
3. Agent selects best ones and formats

Agent Response:

```
I found some programming-related Chuck Norris jokes:

1. "Chuck Norris writes code that optimizes itself."

2. "Chuck Norris doesn't use debuggers. Bugs confess to him."

3. "Chuck Norris can write infinite recursion that terminates."

😎 Developer humor!
```

Example 3: List Categories

User: "What categories are available?"

Agent Process:

1. Calls `get_categories()` tool
2. API returns array of category strings
3. Agent formats list

Agent Response:

```
Chuck Norris jokes are available in these categories:

- animal
- career
- celebrity
- dev
- explicit
- fashion
- food
- history
- money
- movie
- music
- political
- religion
- science
- sport
- travel

Try asking for a joke from any category! For example: "Give me a random dev jo
```

Example 4: Category-Specific Random

User: "Tell me a joke from the dev category"

Agent Process:

1. Calls `get_random_joke(category="dev")` tool
2. API returns random joke from 'dev' category
3. Agent formats joke

Agent Response:

```
Here's a dev joke for you:

"Chuck Norris's keyboard doesn't have a Ctrl key because nothing controls Chuc

🤓 Perfect for developers!
```

How It Works: OpenAPIToolset Deep Dive

1. Specification Structure

```
{
  "paths": {
    "/random": {              // Endpoint path
      "get": {                // HTTP method
        "operationId": "...", // Becomes function name
        "parameters": [...]   // Becomes function args
      }
    }
  }
}
```

2. Auto-Generated Tools

From Spec:

```
{
  "operationId": "search_jokes",
  "parameters": [{ "name": "query", "required": true }]
}
```

ADK Creates:

```
async def search_jokes(query: str) -> Dict:
    """Search for jokes"""
    # ADK handles HTTP request
    response = requests.get(
        "https://api.chucknorris.io/jokes/search",
        params={"query": query}
    )
    return response.json()
```

3. Agent Tool Usage

The Agent constructor accepts toolsets directly - ADK handles the async tool loading internally:

```
root_agent = Agent(
    ...,
    tools=[chuck_norris_toolset]  # Pass toolset directly, not get_tools()
)
```

```
User: "Find jokes about code"
  ↓
Agent (LLM): Decides to call search_jokes
  ↓
search_jokes(query="code") executes
  ↓
HTTP GET https://api.chucknorris.io/jokes/search?query=code
  ↓
API returns: {"total": 5, "result": [...]}
  ↓
Agent (LLM): Formats response for user
  ↓
User sees: "I found 5 jokes about code: ..."
```

4. What ADK Handles Automatically

- ✔️ HTTP request construction
- ✔️ Parameter validation (type, required/optional)
- ✔️ URL building (server + path + query params)
- ✔️ Response parsing (JSON to dict)
- ✔️ Error handling (network, HTTP errors)
- ✔️ Authentication (if specified in spec)

Key Takeaways

1. **OpenAPIToolset = Zero Manual Tool Code**: No need to write `def search_jokes()` yourself
2. **operationId → Function Name**: Controls how LLM sees the tool
3. **parameters → Function Args**: Becomes tool function signature
4. **Works with Any OpenAPI API**: GitHub, Stripe, Twilio, custom APIs
5. **No API Key Needed for Chuck Norris API**: Public and free!

Best Practices

OpenAPI Spec Creation

DO:

- ✔️ Use descriptive `operationId` (e.g., `get_random_joke` not `endpoint1`)
- ✔️ Write clear `description` fields (LLM reads these to decide tool usage)
- ✔️ Mark required parameters correctly
- ✔️ Include response schemas for better error handling

DON'T:

- ❌ Use generic names like `api_call_1`
- ❌ Skip descriptions (LLM won't know when to use tool)
- ❌ Mark all parameters as required (provide sensible defaults)

Tool Design

DO:

- ✔️ One tool per distinct action (get, search, create, update)
- ✔️ Keep parameter lists short (< 5 parameters ideal)
- ✔️ Use enums for categorical parameters
- ✔️ Test tools independently before agent integration

DON'T:

- ❌ Combine unrelated actions in one endpoint
- ❌ Use overly complex nested parameters
- ❌ Assume LLM will infer missing descriptions

Authentication

Chuck Norris API doesn't need auth, but for APIs that do:

```
# API Key in header
OpenAPIToolset(
    spec=spec,
    auth_config={
        "type": "api_key",
        "api_key": os.getenv("API_KEY"),
        "key_name": "X-API-Key",
        "key_location": "header"
    }
)

# Bearer token
OpenAPIToolset(
    spec=spec,
    auth_config={
        "type": "bearer",
        "token": os.getenv("AUTH_TOKEN")
    }
)

# OAuth (more complex, see ADK docs)
```

Common Issues & Troubleshooting

Issue 1: Tool Not Being Called

Problem: Agent doesn't use your OpenAPI tool

Solutions:

1. Check `operationId` is descriptive: `get_random_joke` not `endpoint1`
2. Add detailed `summary` and `description` in spec
3. Test tool directly in Python to verify it works
4. Review agent instruction (does it mention the tool's purpose?)
5. Check Events tab: Is LLM considering the tool?

Issue 2: Import Errors

Problem: `ImportError: cannot import name 'OpenAPIToolset'`

Solutions:

1. Use correct import path: `from google.adk.tools.openapi_tool import OpenAPIToolset`
2. Verify `google-adk` is installed: `pip install google-adk`
3. Check ADK version compatibility

Issue 3: Constructor Parameter Errors

Problem: `TypeError: OpenAPIToolset.__init__() got an unexpected keyword argument 'spec'`

Solutions:

1. Use `spec_dict` parameter instead of `spec`: `OpenAPIToolset(spec_dict=my_spec)`
2. Verify the parameter name in your ADK version

Issue 4: Async Tool Loading Issues

Problem: `ValidationError: Input should be a valid list [type=list_type, input_value=<coroutine object>]`

Solutions:

1. Pass the toolset directly: `tools=[my_toolset]` not `tools=my_toolset.get_tools()`
2. `get_tools()` is async and returns a coroutine - let ADK handle tool loading internally

3. If you need to access tools directly, await the call: `tools = await my_toolset.get_tools()`

Issue 5: Invalid API Response

Problem: Tool returns error or unexpected data

Solutions:

1. Test API endpoint directly with `curl` or Postman
2. Verify spec matches actual API behavior
3. Check required parameters are being passed
4. Look for rate limiting (429 status codes)
5. Validate JSON parsing (use try/except in custom wrappers)

Issue 3: Spec Validation Errors

Problem: ADK rejects OpenAPI spec

Solutions:

1. Validate spec at https://editor.swagger.io/
2. Check OpenAPI version (`3.0.0` or `3.1.0` supported)
3. Verify all required fields present (`openapi`, `info`, `paths`)
4. Use proper JSON types (`string` not `str`, `integer` not `int`)
5. Check for typos in field names

Issue 4: Agent Misinterprets Tool Output

Problem: Agent doesn't properly format API response

Solutions:

1. Improve agent instruction to specify output format
2. Add examples in instruction: "Extract 'value' field from JSON"
3. Use tool result in structured way (dict keys documented)
4. Consider post-processing in custom wrapper function
5. Check response schema in spec matches actual API

Real-World Applications

1. GitHub Integration

Use Case: Code review assistant

OpenAPI Tools:

- `get_pull_request(repo, number)` - Get PR details
- `list_comments(repo, number)` - Get review comments
- `create_comment(repo, number, body)` - Add review comment

Example: "Summarize the changes in PR #123 and check for security issues"

2. Stripe Payment Processing

Use Case: E-commerce support agent

OpenAPI Tools:

- `create_payment_intent(amount, currency)` - Process payment
- `get_customer(id)` - Get customer details
- `create_refund(payment_id, amount)` - Issue refund

Example: "Process a refund of $50 for order #456"

3. Twilio SMS/Voice

Use Case: Communication automation agent

OpenAPI Tools:

- `send_sms(to, body)` - Send text message
- `make_call(to, from, url)` - Initiate phone call
- `get_message_status(sid)` - Check delivery status

Example: "Send a confirmation SMS to customer at +1234567890"

4. Jira Project Management

Use Case: Development workflow agent

OpenAPI Tools:

- `create_issue(project, summary, description)` - Create ticket
- `get_issue(key)` - Get ticket details
- `transition_issue(key, transition_id)` - Move to different status

Example: "Create a bug ticket for the login issue and assign it to the backend team"

Advanced Topics

Custom Response Processing

Sometimes you need to post-process API responses:

```python
from google.adk.tools import OpenAPIToolset

# Create toolset
toolset = OpenAPIToolset(spec=api_spec)

# Wrap with custom processing
async def search_jokes_enhanced(query: str) -> str:
    """Enhanced search with post-processing"""
    result = await toolset.search_jokes(query=query)

    # Extract just the jokes
    jokes = [item['value'] for item in result.get('result', [])]

    # Format nicely
    if not jokes:
        return f"No jokes found for '{query}'"

    return "\n\n".join(f"{i+1}. {joke}" for i, joke in enumerate(jokes[:3]))

# Use enhanced version in agent
root_agent = Agent(
    ...,
    tools=[search_jokes_enhanced]  # Use wrapper instead of raw toolset
)
```

Multiple API Integration

Combine multiple APIs in one agent:

```python
chuck_toolset = OpenAPIToolset(spec=chuck_norris_spec)
github_toolset = OpenAPIToolset(
    spec=github_spec,
    auth_config={"type": "bearer", "token": github_token}
)

root_agent = Agent(
    ...,
    tools=[chuck_toolset, github_toolset, custom_function]
)
```

Rate Limiting Handling

Implement retry logic for rate-limited APIs:

```python
from tenacity import retry, stop_after_attempt, wait_exponential

@retry(
    stop=stop_after_attempt(3),
    wait=wait_exponential(multiplier=1, min=2, max=10)
)
async def api_call_with_retry():
    return await toolset.some_endpoint()
```

Exercises

1. **Add Weather API**: Integrate OpenWeatherMap API with authentication
2. **Build News Agent**: Use NewsAPI to fetch and summarize articles
3. **Create Multi-API Agent**: Combine 3+ different APIs in one agent
4. **Custom Wrapper**: Write post-processing for Chuck Norris API responses
5. **Error Handling**: Add try/except blocks for network failures

Further Reading

- OpenAPI Specification (https://spec.openapis.org/oas/latest.html)
- Chuck Norris API Documentation (https://api.chucknorris.io/)
- ADK OpenAPIToolset Documentation (https://google.github.io/adk-docs/tools/openapi/)
- Swagger Editor (https://editor.swagger.io/) - Test OpenAPI specs
- Public APIs List (https://github.com/public-apis/public-apis) - Find APIs to integrate

Congratulations! You can now connect your agents to any OpenAPI-compliant REST API without writing manual tool code. This opens up integration with thousands of web services!

Next Steps

🚀 **Tutorial 04: Sequential Workflows** - Learn to orchestrate multiple agents in ordered pipelines

---

Generated on 2025-10-21 09:01:53 from 03_openapi_tools.md

Source: Google ADK Training Hub