DTW'24 TM Forum AIVA Hackathon: User Guide

Table of Contents

1 About

The scope of this document is to provide DTW'24 Hackathon participants with enough information to use TM Forum's AI Virtual Assistant (AIVA) API, such that you may work towards the Hackathon's objectives. More context about the Hackathon's and its objectives can be found here.

2 Introduction

Users of TM Forum's rich library of industry knowledge and standards typically face two problems:

1. It takes significant time and effort to identify all the information one is looking for; and
2. It takes additional effort to integrate the TM Forum information with the user's own or other 3rd party information.

TM Forum AIVA is an enterprise search agent built with Google Cloud Gemini and the Vertex AI Agent Builder. It addresses the first problem above, by using a natural language Q&A interface, to rapidly obtain relevant content from TM Forum's rich industry knowledge & standards library. In this hackathon, we provide you with TM Forum AIVA API to address the second problem.

TM Forum AIVA API offers you two functions:

1. retrieving applicable information assets from relevant sources in TM Fourm's knowledge base (web text, images and PDF documents); and/or
2. generating server stub or client code, or documentation for ODA API specs.

Google Cloud's LangChain on Vertex AI provides a reasoning engine that:

1. allows you to ask a natural language question,
2. detects whether intent is to obtain some information or to generate code,
3. calls the right back-end function
4. and provides a summary from the information obtained from the functions.

(Hint: you may want to think about wrapping up TM Forum AIVA API and your own enterprise APIs as functions that could be accessed seamlessly using the same reasoning engine architecture used to build TM Forum AIVA API).

2.1 Example Use Case

2.1.1 Scenario

You are a Principal Architect for NWOps Innovation. You want to develop a proposal to convince your leadership to invest in exploring the use of Knowledge Graphs for achieving autonomous networks.

2.1.2 Information Requirements

- Has anybody operationalized Knowledge Graphs for NWOps Innovation?
- For which particular use cases?
- Which problems did they solve, or which benefits did they realise?
- How important are these problems/ benefits for my own business?

2.1.3 Answering Industry Questions

We can see on the next page how TM Forum's website has excellent answers for the first three questions above, helping you to make a strong case for your proposal. Towards the end of this guide, we will revisit how you could integrate this information with your own enterprise information repository. First, we will look at how TM Forum AIVA works.

Query: “Describe case studies that use AI knowledge graphs for autonomous networks including use cases and benefits.”

TM Forum AIVA Response (after Markdown rendering)

Case Studies of Knowledge Graphs for Autonomous Networks:

Here are some case studies of how knowledge graphs are being used to improve autonomous networks:

China Unicom:

- **Use Case:** China Unicom deployed an AI knowledge graph to automate network fault management across multiple mobile domains.
- **Benefits:**
  - **Increased Efficiency:** Fault handling efficiency improved by 67%, reducing the average process time from 45 minutes to 15 minutes.
  - **Cost Reduction:** Annual costs decreased by 11.3 million euros due to reduced manual analysis.
  - **Improved Accuracy:** Root Cause Analysis (RCA) accuracy increased from 70% to 89%.
  - **Knowledge Sharing:** Knowledge gathering increased by 38%, and knowledge dissemination time was reduced from 2 months to 2-4 weeks.
  - **Interpretability:** 100% of the inference processes are now interpretable, aiding front-line engineers in understanding recommended actions.

China Telecom:

- **Use Case:** China Telecom is using a knowledge graph in conjunction with a large language model (LLM) to improve network autonomy.
- **Benefits:**
  - **Improved Production Efficiency:** Production efficiency increased by 50% compared to manual processes.
  - **Enhanced Fault Identification:** The knowledge graph helps identify causal relationships between separate network faults, improving prediction accuracy.
  - **Knowledge Retrieval and Reasoning:** The knowledge graph enables knowledge retrieval, reasoning, and intelligent question and answer capabilities.
  - **Interpretability:** The knowledge graph provides interpretable inference processes, ensuring transparency in decision-making.

Green AI for Autonomous Networks:

- **Use Case:** This Catalyst project focuses on building autonomous knowledge graphs by sharing and applying vertical domain knowledge graphs across network topologies.
- **Benefits:**
  - **Energy Savings:** The network knowledge graph enables real-time analysis of network parameter indicators, leading to energy-efficient collaborative control mechanisms and optimization strategies.
  - **Continuous Optimization:** Deep learning and big data analysis are used to continuously optimize the network based on data from base station KPIs, alarms, cases, work orders, and business information.

3 Getting Started

3.1 Pre-requisites

- You can call the API using any language of your choice that supports RESTful interfaces (e.g., Python, Java, Golang,...). You can also use Google's Vertex AI SDK.
- You will require your own GCP project (corporate or personal), TM Forum will then provide the associated account access to the reasoning agent. This will be followed by confirmation from TM Forum for specific users, along with provision of API key credentials.
- TM Forum AIVA API Agent (link tbc) is provided as a Google Cloud LangChain on Vertex AI Reasoning Engine endpoint.

3.2 Authentication

Once users have obtained access to TM Forum AIVA service, they can use the code given in the notebook to authenticate and access AIVA from their own GCP project or web browser by using the access token obtained from TM Forum.

4 Using TM Forum AIVA API

As mentioned earlier, you can submit a query to the API to 1) seek information or to 2) generate ODA API code. The following personas who may benefit from these capabilities are the following:

1. Anyone who needs an introduction to specific areas within TM Forum content (e.g., ODF/ODA, Catalysts, research & analysis,...);
2. Strategic decision-makers who prioritize and plan key initiatives, such as sustainability;
3. Network/software architects and developers who wish to identify the TM Forum specifications relevant to their domain (e.g., billing), and use automatic generators to accelerate their deployment/modernization journeys.

4.1 Example Questions

Below are some example questions of either type:

4.1.1 Information-seeking

- Describe ODF
- Describe how AI-driven Telco is different from Digital Telco and Traditional Telco. What are some of the key points of difference?
- Describe case studies that use AI knowledge graphs for autonomous networks including use case and benefits
- How do I test my level of autonomous operations?
- How do I start building an autonomous network?
- Is ODA AI-ready?

4.1.2 ODA API Code Generation

- Generate code in Java for a product catalogue
- Generate code in Python Flask for a Trouble Ticket API
- Generate code for TMF637

4.2 Web, Image and PDF Search

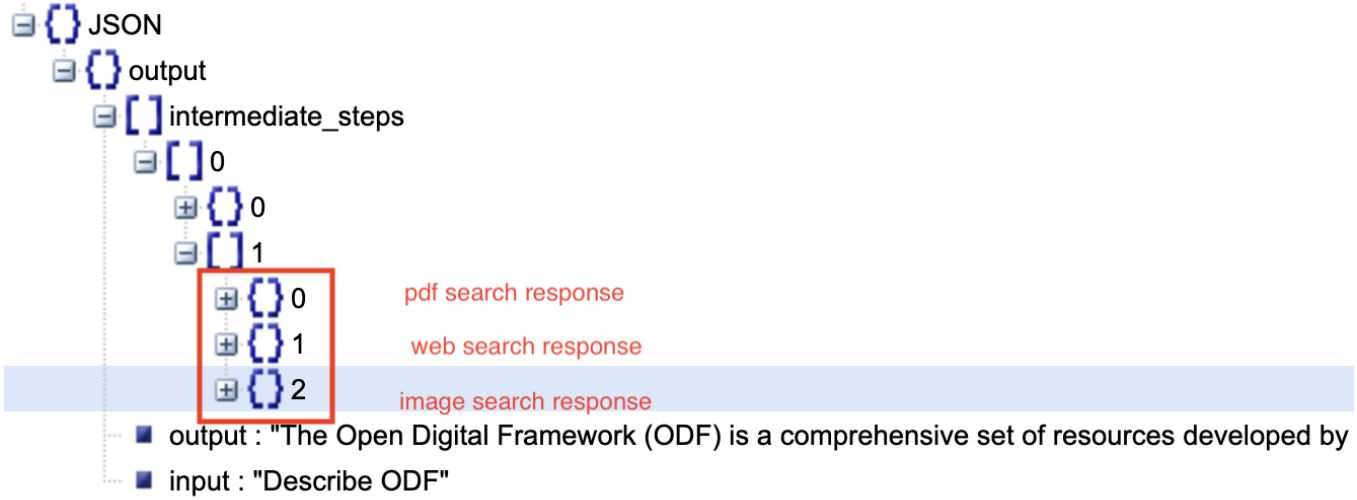
4.2.1 Functionality Overview

This feature searches for TM Forum content (webpages, images and PDF documents) which is relevant to the user's query. The content is gathered from the text and image embeddings created by the Vertex AI Agent Builder.

Here is the payload/data format in which a query, "Describe ODF", would be submitted to the API:

```
{
    "input": {
      "input": "Describe ODF"
    }
}
```

Here is a high-level view of the JSON format in which the API output is received:



[IMPORTANT NOTE: All search output will NOT have all the three Search responses. Certain responses, for example, may not have any relevant images identified. Please see an example parser of the response that shows how to check for such possibilities in TM Forum AIVA Hackathon-User-Guide.ipynb notebook.]

Please refer to the appending information in Hackathon Webinar.pptx.pdf for additional images showing further details about the response JSON. The following are the full JSON request and response files:

1. search_request.json
2. reasoning_agent_search_response.json

4.2.2 API Response Processing Examples

Here are a few example Python statements to print some specific elements in the response JSON whose structure we saw in the last section ('res' represents the full response JSON):

```
#The final summary from the agent
print(res['output']['output'])

#The summary from searching the PDF documents
print(res['output']['intermediate_steps'][0][1][0]['answer']['answerText'])

#The summary from searching the web site content
print(res['output']['intermediate_steps'][0][1][1]['answer']['answerText'])

#The title of the first reference from searching the PDF documents
print(res['output']['intermediate_steps'][0][1][0]['answer']['references'][0]['chunkInfo']['documentMetad
ata']['title'])

#The document that contains the first image that was retrieved as relevant to the query
print(res['output']['intermediate_steps'][0][1][2]['results'][0]['document']['derivedStructData']['image'
]['contextLink'])
```

As mentioned earlier, all responses may not contain all these elements. Please see TM Forum AIVA Hackathon-User-Guide.ipynb for an example of how to write a more robust parser of the output.

4.3 Swagger ODA Code Generation

4.3.1 Functionality Overview

This feature initially takes a user's natural language query and generates either a stub server, client code or documentation based on a Swagger specification file in JSON or YAML format. If there are multiple versions of a specification file, it chooses to generate code for the latest version as determined by only the version number (it does not consider a specification's date or other metadata).

Important Note: The code generator determines if it is going to generate server or client code, or documentation based on the language specified in the prompt requesting code generation. Here are the languages categorized by the type of code or documentation they are used for:

Client	Server	Documentation
csharp	aspnetcore	dynamic-html
csharp-dotnet2	go-server	html2
dart	java-vertx	
go	nodejs-server	
java	python-flask	
javascript	spring	
php		
python		
r		
ruby		

NOTE: You can use part of a language identifier string listed above. For example, you can just say *"generate code in ASP"*, and it will map it to ASP.NET Core. However, currently, it does not handle ASP.NET.

4.3.2 API Response Processing Examples

The prompt and the response JSON formats for code generation are shown in the following files:

1. code_gen_request.json
2. reasoning_agent_code_gen_response.json

The response from the code generator is simpler than the one from Search. The following Python statement would obtain the final output:

```
#The final summary from the agent
print(res['output']['output'])
```

4.4 Useful Tools

Tools which can read JSON-based REST API calls such as Postman or JSONView will also benefit you in building your hackathon solution.

4.5 Troubleshooting

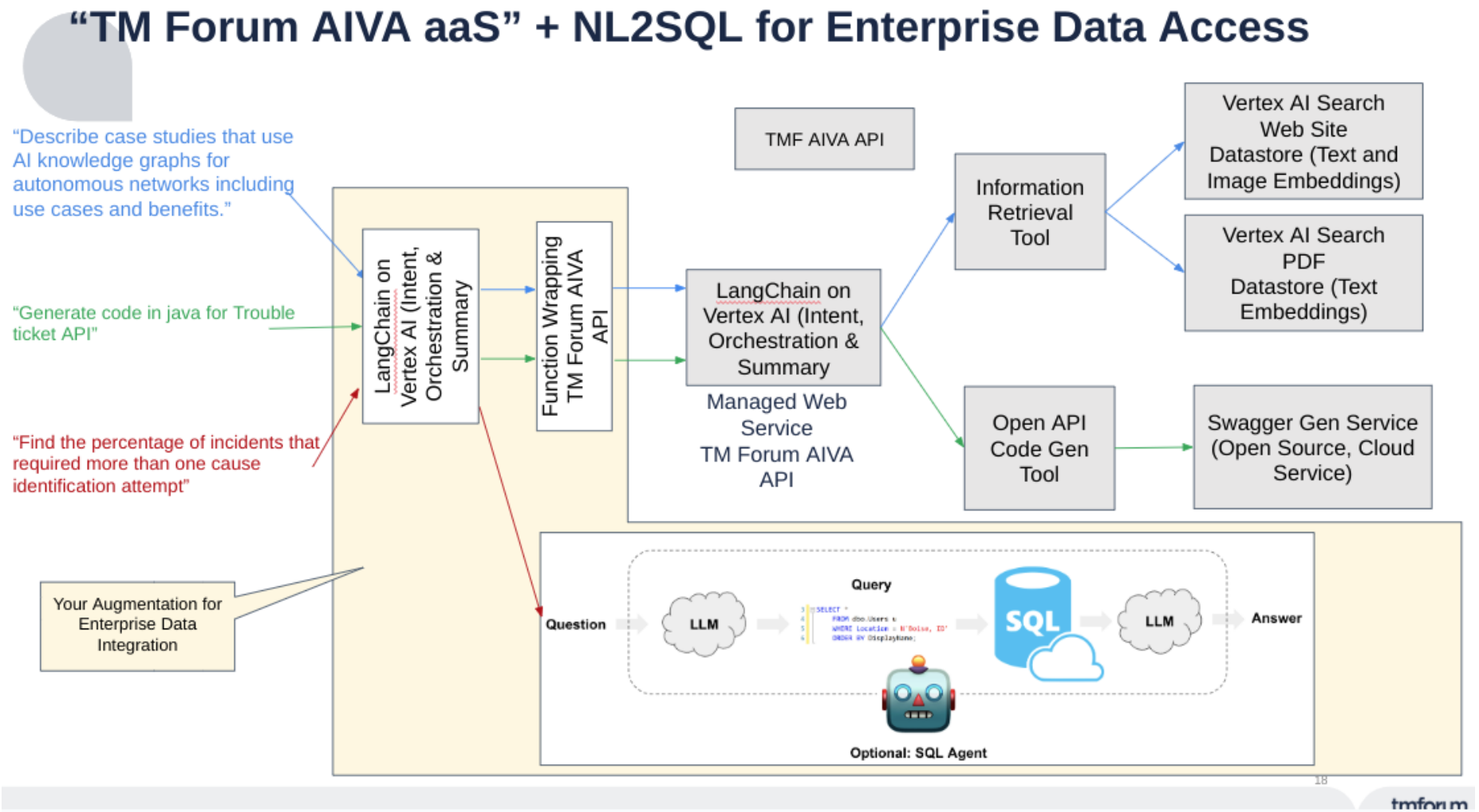
To provide technical support we have a dedicated Hackathon Support team available in person to assist you. However, your first point of contact should be a Google Mentor.

5 Accessing Internal Knowledge Repositories

In Section 2's detailed example, we discussed how developing a proposal for exploring the use of AI Knowledge Graphs for autonomous networks needs information on both the industry success stories and internal challenges or opportunities. We also saw how the TM Forum AIVA API makes it easy to find highly relevant industry information. In this section, we will explore how we could use Google Cloud Gen AI capabilities to query your own enterprise data on network operations using natural language and integrate it seamlessly with the TM Forum AIVA. This requires two steps:

1. Develop a NL2SQL capability using Gemini; and
2. Use the same Google Cloud LangChain or Vertex AI Reasoning Engine used in the development of the AIVA API to integrate this NL2SQL capability with the AIVA API with automated intent detection, orchestration and summarization.

The diagram on the following page describes how such is implemented.



Please see the Colab notebook, TM Forum AIVA Client Plus NL DB Access.ipynb for an implementation of the above architecture. It will give you an idea of how to use Gen AI for building nested agents for adding functionality to integrate information from multiple internal or external sources. If you are looking for a no-code solution for integrating functionality using Gen AI, please look at Vertex AI Agents.