Spotfire Copilot Orchestrator endpoint

I was curious about
Spotfire Copilot has an endpoint that is called /orchestrator.
The Example request body for this endpoint is as follows.

Orchestrator endpoint body

```
 {
  "request_tag": "OrchRequest",
  "temperature": 0,
  "index_name": "string",
  "index_score_threshold": 0,
  "index_topk": 0,
  "retriever_type": "string",
  "use_secondary_model_plugin": true,
  "llm_name": "string",
  "llm_mode": "string",
  "system_prompt": "Your are an AI assistant. Answer the question based on
the context below. Keep the answer short and concise.",
  "user_intent": "QueryIntent",
  "history": "[{'role': 'user', 'content': 'question1'}, {'role':
'assistant', 'content': 'response1'},{'role': 'user', 'content':
'question2'}, {'role': 'assistant', 'content': 'response2'}]",
  "image": "string",
  "image_url": "string",
  "client_data": [
    {
      "data_name": "string",
      "value": "string"
    }
  ],
  "prompt": "How to create a bar chart?"
}
```

With this example we can construct a body to send to the orchestrator endpoint to manipulate the model and/or intent to use.
It also allows to specify more granular controls like setting the system prompt and other elements like temperature. The Orchestrator uses this information to route requests to you LLM provider or vector databases. to generate responses.

Proof of Concept

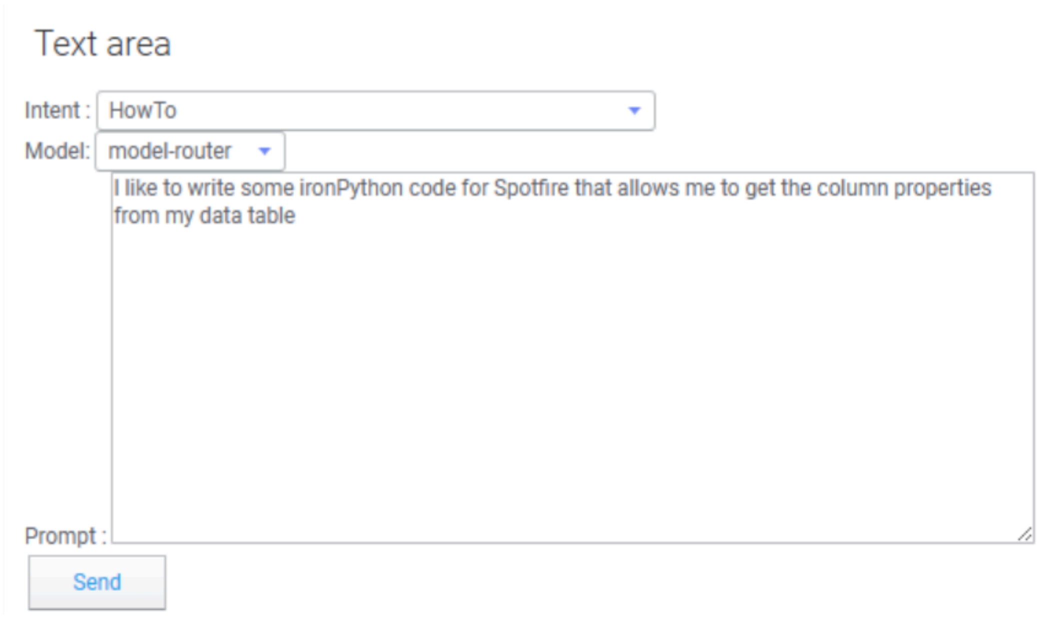
I wanted to prove its possible to do both intent and model switches with Spotfire Copilot
I first started off with a Python data function that calls the endpoint and sends in the body.
e.g.:

```
{
  "llm_name": "gpt-4.1-mini",
  "user_intent": "QueryIntent",
  "prompt": "How to create a bar chart?",
}
```

This returns an answer from the Orchestrator endpoint, and I can see in the backend of the Azure monitoring that it is actually using the specified model.

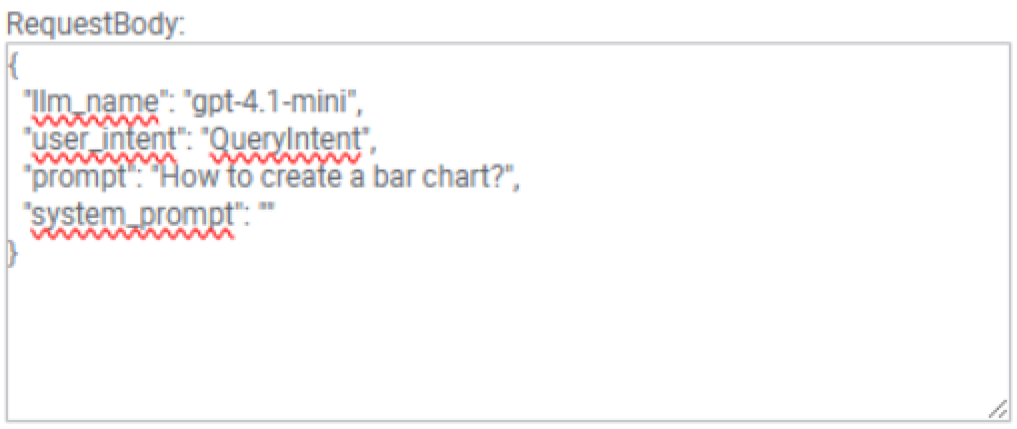
Spotfire in the mix

In a text area I created two fixed value drop downs, one with the registered intents and one with models I have registered in Azure Foundry.
Of course I want to be able to ask different questions so I also added a prompt in a multi line input property.



Intent : HowTo
Model: model-router
Prompt : I like to write some ironPython code for Spotfire that allows me to get the column properties from my data table

Adding a variable request body allows me to manipulate the scaffolding to use. below is a quite minimal settings but with one powerful addition, system_prompt



RequestBody:
```
{
 "llm_name": "gpt-4.1-mini",
 "user_intent": "QueryIntent",
 "prompt": "How to create a bar chart?",
 "system_prompt": ""
}
```

```
{
  "llm_name": "gpt-4.1-mini",
  "user_intent": "QueryIntent",
  "prompt": "How to create a bar chart?",
  "system_prompt": ""
}
```

Adding an alternative system prompt

Example system prompt I generated with another AI

```
You are an expert in data visualization and data analysis. You specialize in
helping users perform quick and effective visual analyses using Spotfire.
Your role is to assist with tasks like data cleaning, duplicate removal,
numerical formatting, and writing IronPython scripts within Spotfire. When
given a user request, provide clear, concise, and actionable steps or code
snippets as needed. Always ensure your responses are tailored to the
specific requirements of the user and align with best practices in data
analysis."

Example Use Cases:

Data Cleanup:
"I have some messy data with missing values and inconsistencies. Can you
help me clean it up?"
Deduplication:
"There are duplicate entries in my dataset that I need to remove. How can I
do this efficiently?"
Number Formatting:
"I want to format the numbers in my table to show as percentages instead of
decimals."
IronPython Code:
"I need to write an IronPython script to automate some data processing tasks
in Spotfire. Can you provide a sample script for this?"
Response Format:

For each request, provide step-by-step instructions or code snippets.
Use clear language and avoid jargon unless necessary.
Highlight any assumptions made about the data or context.
```

While working on the code I found I can manipulate the given request body through direct json key changes like json_data["user_intent"] = user_intent which is a data function input mapped to the provided dropdown.
Below is the complete Python code allowing do do all this

```
import requests
import json
url = "http://yourcopilot.orchestratorURL/orchestrator"  # Replace with your API endpoint
token = "YourToken"         # Replace with your actual token
headers = {
    "Authorization": f"Bearer {token}",
    "Content-Type": "application/json"  # Adjust content type as needed
}

json_data = json.loads(copilot_requestBody) #convert string into JSON object
json_data["user_intent"] = user_intent
json_data["llm_name"] = llm_model_name
json_data["prompt"] = user_input  # Update the prompt field
json_data["system_prompt"] = f"{system_prompt}"

data_json = json.dumps(json_data, indent=4)
response = requests.post(url, headers=headers, json=json_data)

print("Status Code:", response.status_code)

Copilot_resultDocProp = json.loads(response.content.decode('utf-8'))["result"] + '\n request body: \n' + data_json  # returning in a multiline input property including the sent request
Copilot_resultDecoded = json.loads(response.content.decode('utf-8'))["result"]
```

The answer returned for above sample question is as follows

```
Great idea. Here are two solid options you can drop into Spotfire IronPython
to get the properties of columns from a data table. The first is a
straightforward listing of a few common properties. The second uses
reflection so you can see all public properties without hard-coding names.

Assumptions
- You know the exact data table name (replace "YourTableName" with your
table name).
- You want to see what properties exist on the DataColumn objects (not all
properties are guaranteed to exist in every Spotfire version).

Option 1: Quick view of common properties (no reflection)
- Pros: Simple and fast.
- Cons: You must know the properties you want to inspect.

Code:
# IronPython script to print common column properties for a data table
tableName = "YourTableName"  # <-- replace with your table name
```

```
dt = Document.Data.Tables[tableName]

for col in dt.Columns:
    print("Column: {0}".format(col.Name))
    # Try a few commonly available properties (names may vary by version)
    try:
        print("  DataType: {0}".format(col.DataType))
    except Exception:
        pass
    try:
        print("  Type: {0}".format(col.Type))
    except Exception:
        pass
    try:
        print("  IsKey (primary key): {0}".format(getattr(col, "IsKey",
None)))
    except Exception:
        pass
    try:
        print("  IsNullable: {0}".format(getattr(col, "IsNullable", None)))
    except Exception:
        pass
    try:
        print("  Size: {0}".format(getattr(col, "Size", None)))
    except Exception:
        pass
    print()

Option 2: Discover all public properties via reflection (recommended for
future-proofing)
- Pros: No need to guess property names; you’ll see whatever the API
exposes.
- Cons: More verbose output; allows you to adapt to what exists in your
Spotfire version.

Code:
# IronPython script to enumerate all public properties of each column using
reflection
import clr
clr.AddReference("System")
from System.Reflection import BindingFlags

tableName = "YourTableName"  # <-- replace with your table name
dt = Document.Data.Tables[tableName]

for col in dt.Columns:
    print("Column: {0}".format(col.Name))
    t = col.GetType()
    for prop in t.GetProperties(BindingFlags.Public |
BindingFlags.Instance):
        # Skip the obvious column identity to reduce noise
        if prop.Name in ("Name", "Parent"):
            continue
        try:
            value = prop.GetValue(col, None)
        except Exception:
            value = None
        print("  {0}: {1}".format(prop.Name, value))
    print()
```

Optional: export to CSV for easier inspection
If you want to export the results to a CSV for sharing or documentation, you can write to a file using System.IO. Note: file I/O in Spotfire scripts may be restricted by your security settings; if allowed, this version writes one line per column with its key properties.

Code snippet (example, adapt path as needed):

```
import clr
clr.AddReference("System")
from System.IO import StreamWriter, File
from System.Text import Encoding

tableName = "YourTableName"  # <-- replace
dt = Document.Data.Tables[tableName]

path = r"C:\Temp\ColumnProperties.csv"  # ensure the path exists and you
have write permission
with StreamWriter(path, False, Encoding.UTF8) as writer:
    writer.WriteLine("ColumnName,Property,Value")
    for col in dt.Columns:
        t = col.GetType()
        for prop in t.GetProperties(BindingFlags.Public |
BindingFlags.Instance):
            try:
                value = prop.GetValue(col, None)
            except:
                value = None
            writer.WriteLine("{0},{1},{2}".format(col.Name, prop.Name,
value))
```

Tips and best practices
- Replace "YourTableName" with the actual name of the data table you’re inspecting.
- If you rely on many properties, the reflection approach (Option 2) is safer across Spotfire versions.
- If you want to filter columns (e.g., only numeric columns), add a quick check on col.DataType or other properties before printing.
- Run the script from Tools > Scripting in Spotfire (IronPython). The output usually appears in the Script Output pane or the console depending on your

```
environment.

If you share your Spotfire version and the exact properties you’re most
interested in (e.g., DataType, IsNullable, etc.), I can tailor a compact
script that prints or exports just those fields.
 request body:
{
    "llm_name": "model-router",
    "user_intent": "HowTo",
    "prompt": "I like to write some ironPython code for Spotfire that allows
me to get the column properties  from my data table",
    "system_prompt": "You are an expert in data visualization and data
analysis. You specialize in helping users perform quick and effective visual
analyses using Spotfire. Your role is to assist with tasks like data
cleaning, duplicate removal, numerical formatting, and writing IronPython
scripts within Spotfire. When given a user request, provide clear, concise,
and actionable steps or code snippets as needed. Always ensure your
responses are tailored to the specific requirements of the user and align
with best practices in data analysis.\"\n\nExample Use Cases:\n\nData
Cleanup:\n\"I have some messy data with missing values and inconsistencies.
Can you help me clean it up?\"\nDeduplication:\n\"There are duplicate
entries in my dataset that I need to remove. How can I do this efficiently?
\"\nNumber Formatting:\n\"I want to format the numbers in my table to show
as percentages instead of decimals.\"\nIronPython Code:\n\"I need to write
an IronPython script to automate some data processing tasks in Spotfire. Can
you provide a sample script for this?\"\nResponse Format:\n\nFor each
request, provide step-by-step instructions or code snippets.\nUse clear
language and avoid jargon unless necessary.\nHighlight any assumptions made
about the data or context."
}
```