Technical Evaluation
Mulesoft API Specification Design & Implementation
By Diego Perez Villa for SPS Tech Is Now
July 27, 2023

Introduction
The purpose of this documentation is to act as a comprehensive step-by-step guide for the evaluators of this technical practice. It will contain code snippets alongside an explanation of how it works and why it was implemented.

It is divided into two sections: API Specification Design, using Anypoint Design Center and Mulesoft API Implementation, using Anypoint Platform. The repository for this code can be found in the email this file was sent or at [this](#)link.

Technical Evaluation Overview
Star Wars API

(taken from Technical Evaluation)

Context: A Star Wars Universe Compendium mobile app is being developed and will require an API to retrieve character information.

Requirements: Information about all Star Wars characters must be retrieved in CSV format, with an additional optional query parameter to retrieve all characters of a specified gender.

API Specification Design

In order to create the API Specification, Anypoint Design Center was used. The following is the API spec in RAML format:

star-wars-api.raml

```
#%RAML 1.0
title: Star Wars API
version: v1
description: An API for Star Wars Characters
mediaType: application/csv


/characters:
```

```
  get:
    queryParameters:
      gender:
        type: string
        description: use to query characters by gender
        example: female
        required: false


    responses:
      200:
        body:
          application/csv:
            type: string
            example: !include /characters.csv
```

At the root, there is a title, version, description and MediaType.
BaseUri, Types can also be provided, although they have been omitted
given the evaluation's minimal complexity.

For the resources, there is only one: characters, with a single get
method and its optional query parameter. A nested resource would not
fit well here in comparison.

By using Anypoint API Design Center, a mock server can be used to
validate the API's functionality.

API Implementation in Anypoint Studio

Once the API Specification has been created, it can be used to design the Mulesoft application in Anypoint Studio.
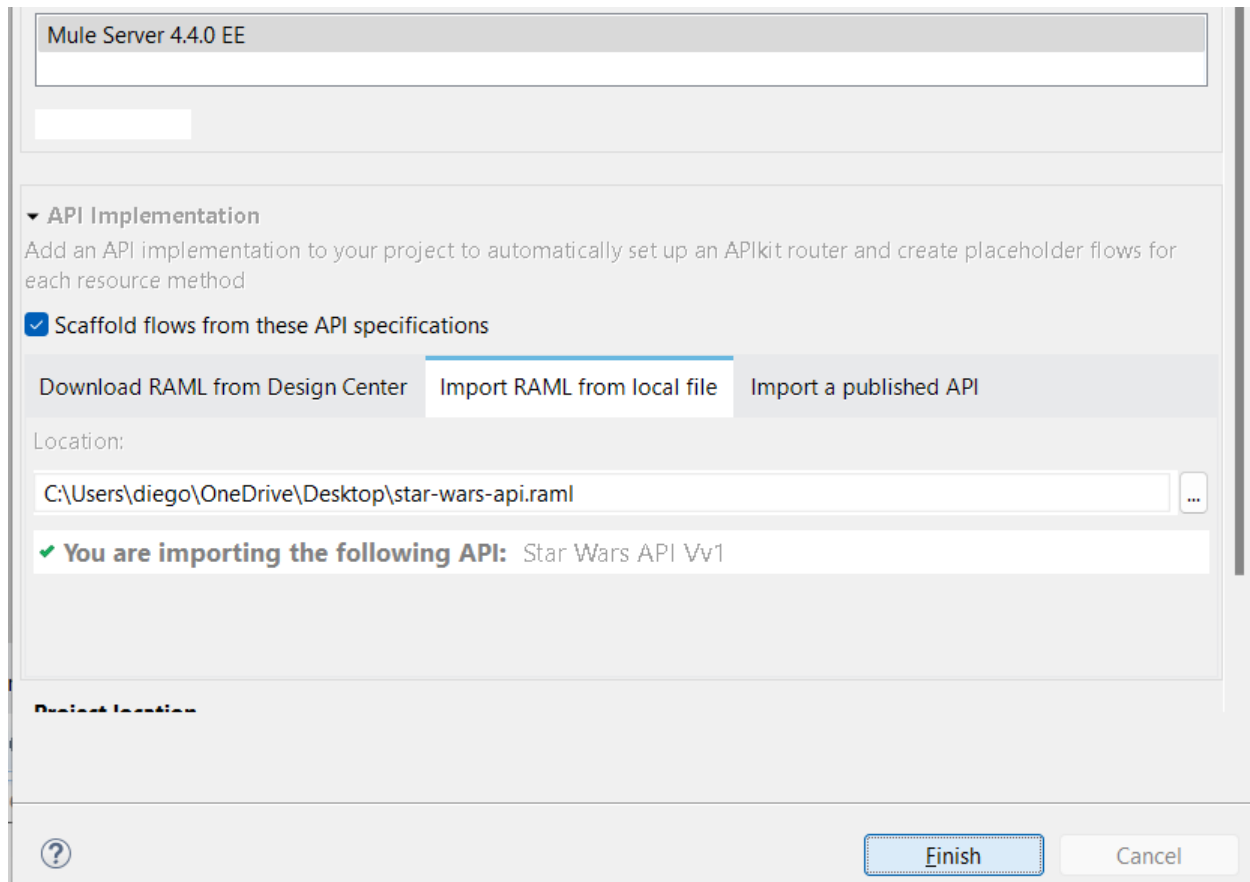
The main advantage this approach provides is the scaffold flows, which include the following:

star-wars-api-main
This flow is comprised of two components, an HTTP Listener and an APIKit Router, as well as a robust Error Handling system, which provides an **On Error Propagate** for:
   - APIKIT:
        - BAD REQUEST
        - NOT FOUND
        - METHOD NOT ALLOWED
        - NOT ACCEPTABLE
        - UNSUPPORTED MEDIA TYPE
        - NOT IMPLEMENTED
        -
star-wars-api-console

This flow is comprised of an HTTP Listener and the APIKit Console, which is a resource that grants developers access to the API documentation, and provides an experience similar to that of using API Designer

get:\characters:star-wars-api-config
This flow acts as the get method's resource location. In it is a single transform component, with an inline Dataweave code that writes the example csv text to the payload.
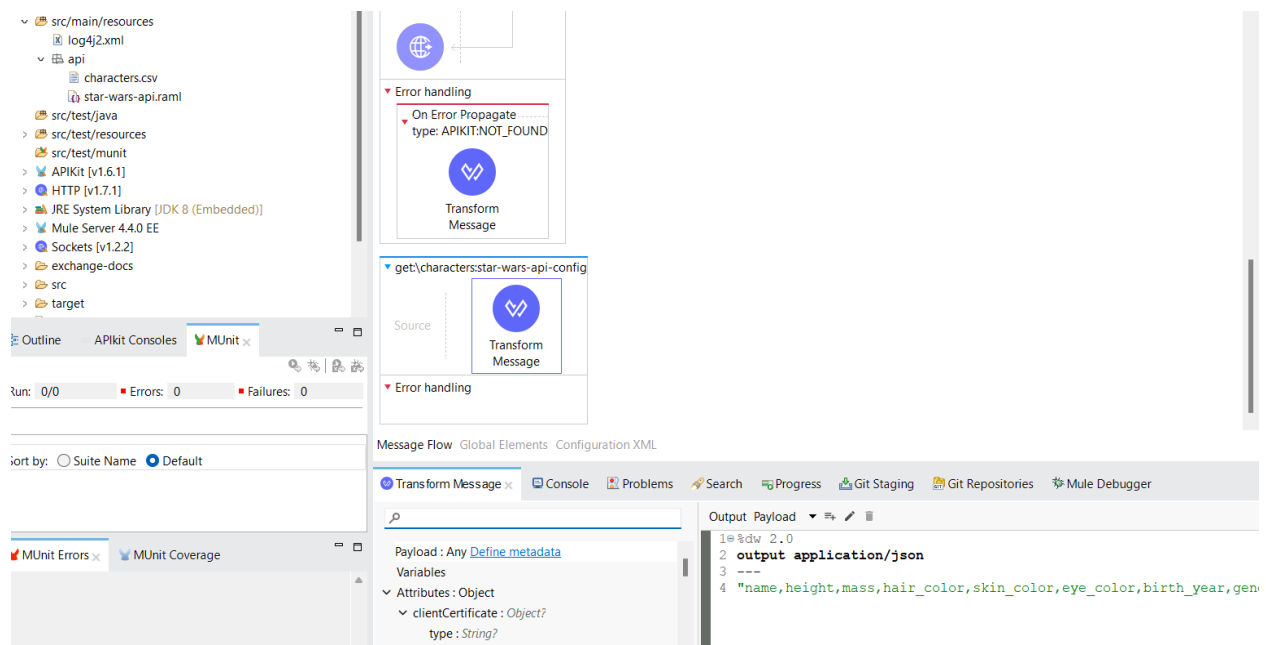
On the top left, inside src/main/resources/api, the characters.csv and the RAML specification can be found. The same content within characters.csv is written to the payload by the Transform Message component highlighted in blue, inside the get:\characters:star-wars-api-config flow.

Additionally, 2 global configuration elements are created. These are required for the listeners and APIKit components to work properly, and contain information pertaining to the host, port, and RAML files.

Running the Application

Up to this point, the Mulesoft Application has been generated on its own, using the RAML as a base to generate the scaffold flows. However, it can be tested, and it will perform the same as the mock server in the API designer.

The application is run by left-clicking the star-wars-api whitespace -> run project starwars-api. The console will indicate when it has been deployed.

The global element's HTTP listener configuration port is by default set to 8081. Through a browser or an API platform such as Postman, the following url can be accessed:

Localhost:8081

The following message will appear:

No listener for endpoint: /

In the star-wars-api-main -> Listener component -> General -> Path, it is specified that the endpoints will be located at:

Localhost:8081/api/characters

The *characters* resource will need to be added as well, without it, the application will return the following error: "Resource not found". Once we access the API, we can reach the payload message set in the **get:\characters:star-wars-api-config** flow.
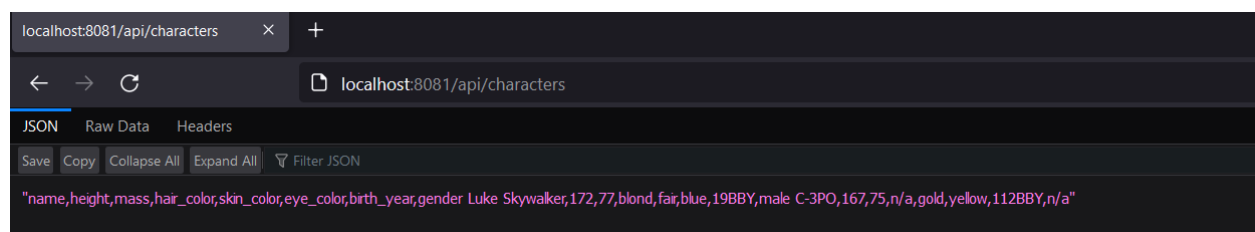

Modifying the Mule Project

Now that the functionality has been tested, it is important to modify the Mule project so that it can retrieve information from the Star Wars API, therefore acting as an intermediary between the user and the API itself.

Notably, the query parameter has not been considered until this point. However, it will be important to retrieve it when querying the Star Wars API. Additionally, the Mule application's MIME type must be application/csv, as it is currently application/json.

Connecting to SWAPI

In order to connect to SWAPI, it is necessary to create a Request Component at the start of the **get:\characters:star-wars-api-config** flow.
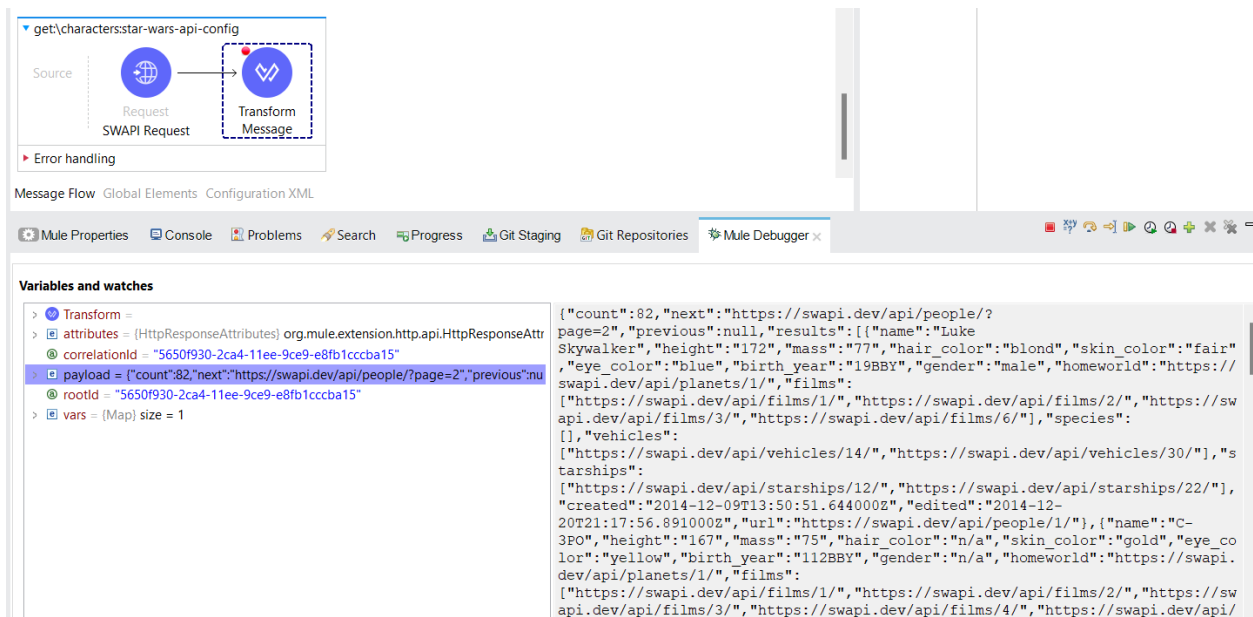
Here, the application has been run in debug mode, and the Mule Debugger shows the contents of the payload after the SWAPI request. It is a JSON that contains a fraction of the results. In order to obtain the entirety of the characters, it will be necessary to query the SWAPI multiple times, and store the results.

Obtaining the Specified Fields

In order to obtain all the fields, two things must be done:

1. Extract all data from the SWAPI database. One query will only return 10 character entries or less, and it is necessary to traverse the entirety of the SWAPI's character pages in order to retrieve all characters. This can be done by either checking the "next" field in the SWAPI response or by determining the page range necessary to obtain all characters.

2. Extract the specified fields from the payload. This can be done by taking the payload result and using Dataweave playground to quickly obtain the desired dataweave code.

As seen above, the Dataweave code is a simple extraction of the first 8 values of each Star Wars character. This output will then be stored and added to the subsequent outputs, until the character list is exhausted.

Accumulating Results

The SWAPI returns 10 characters at a time, as well as a count of how many characters it contains in the database. Therefore, by dividing the count by 10, the number of pages to query can be determined, in order to generate an array to iterate over in a For Each Component.
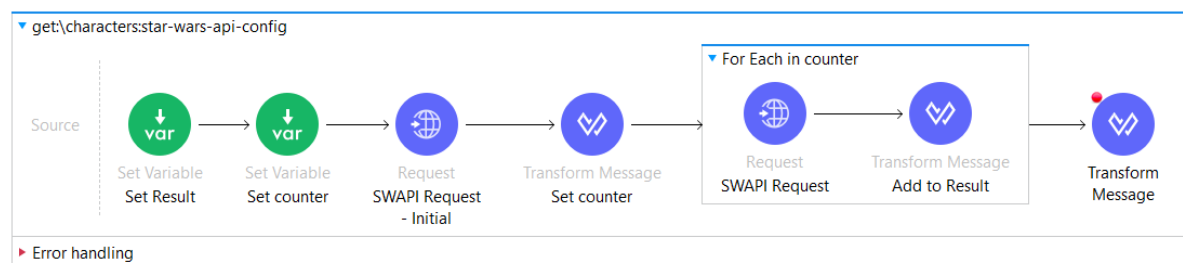

Above is the altered flow. Two variables have been set, a Result array and a counter integer. In the initial SWAPI request, the

character count is obtained. It is then divided by ten, and the ceiling of that result is used in the For Each counter. In the counter, the flow iterates over the SWAPI character pages, and stores the results in the result array.

Up to this point, the flow will query the SWAPI multiple times and store the result as a JSON array. The two remaining steps are to filter the results if a query parameter is provided, and to convert the JSON array to a CSV.

The first task is simple: a new variable (genderFilter) for the query parameter must be created, in order to preserve the query parameter attribute after the For Each.
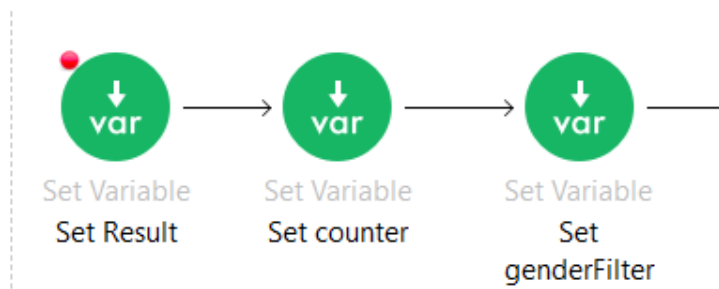
In order to filter the final result, it is possible to use a Choice component or a single Dataweave transform component with an if statement. In this example, a Choice component evaluates the result of the query Parameter, if it is null it will perform the following filter and return the payload in CSV format, if not it will only return the payload converted to CSV.

```
%dw 2.0
output application/csv quoteValues=false,header=true
---
vars.result filter $.gender == vars.genderFilter
```
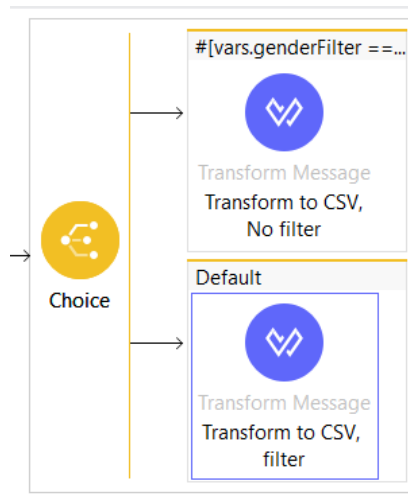
With this flow, the mule application is now able to query all Star Wars characters from SWAPI, and filter the results by gender, if necessary. If the mulesoft application is tested via browser, the CSV will get automatically downloaded.

Closing Remarks

Overall, the concept of this mule application is quite simple. However, there are certain limitations that may impact performance if this application were to be used in a production environment. The main one being the limit on the amount of queryable characters by SWAPI.

Notes from the Author

I was given approximately 2 days to work on this project, of which I've only had about 4-5 hours of free time. If I had had more free time, perhaps I would have worked on MUnits, and made the querying process more efficient. I tried a parallel for Each, but that probably would not have worked, at least not with the array iteration approach.

I'm more used to starting a Mulesoft project from scratch in Anypoint studio, although API Designer is very useful when it comes to API implementation. Moving forwards, I'd like to work with it a lot more, as I feel as though the API Specification came out a bit barebones.