Global Weather Project

Challenges – Design/development Version 1.0

Table of Contents

[Purpose of this document 2](#_Toc31016609)

[Challenges along with their solutions. 2](#_Toc31016610)

[Design Decisions 4](#_Toc31016611)

[Exception Handling approach 5](#_Toc31016612)

[Modularity and maintainability 7](#_Toc31016613)

# Purpose of this document

This document is about detailed information about challenges that were faced during design and development of the solution along with troubleshooting and solutions. It also covers the other aspects such as design decisions, exception handling approach and the modularity and maintainability approach taken.

# Challenges along with their solutions

**Challenge1: Webservice ‘http://www.webservicex.com/globalweather.asmx?WSDL’ and dockerized version of the service seems down/not working.**

The globalweather SOAP based service has two operations in it.

1. **GetWeather** : This operation expects a country name and returns the weather details for that specific country.
2. **GetCitiesByCountry**: This operation expects a country name and returns the cities of that specific country.

**Solution**:

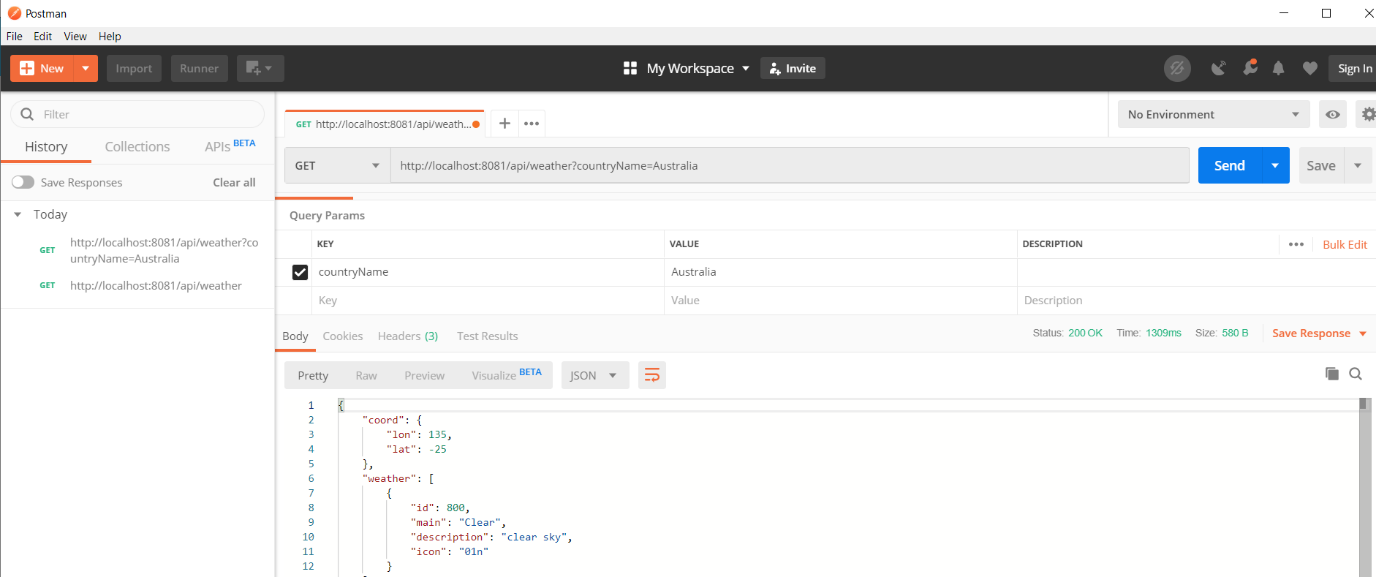
Since the original webservice (SOAP based) was not working, I found couple of REST based webservices that can be utilized for exposing these operations in REST format.

Below are these webservices.

* <https://openweathermap.org>

This webservice expects an APPID as a header value which is provided as part of subscription along with the country name.

e.g. Weather details for Australia as below as seen below in postman



In order to overcome the issue with the original webservice, I used this webservice to retrieve weather details by passing one of the countries from configured countries 'Australia','India','France','New Zealand','Singapore'

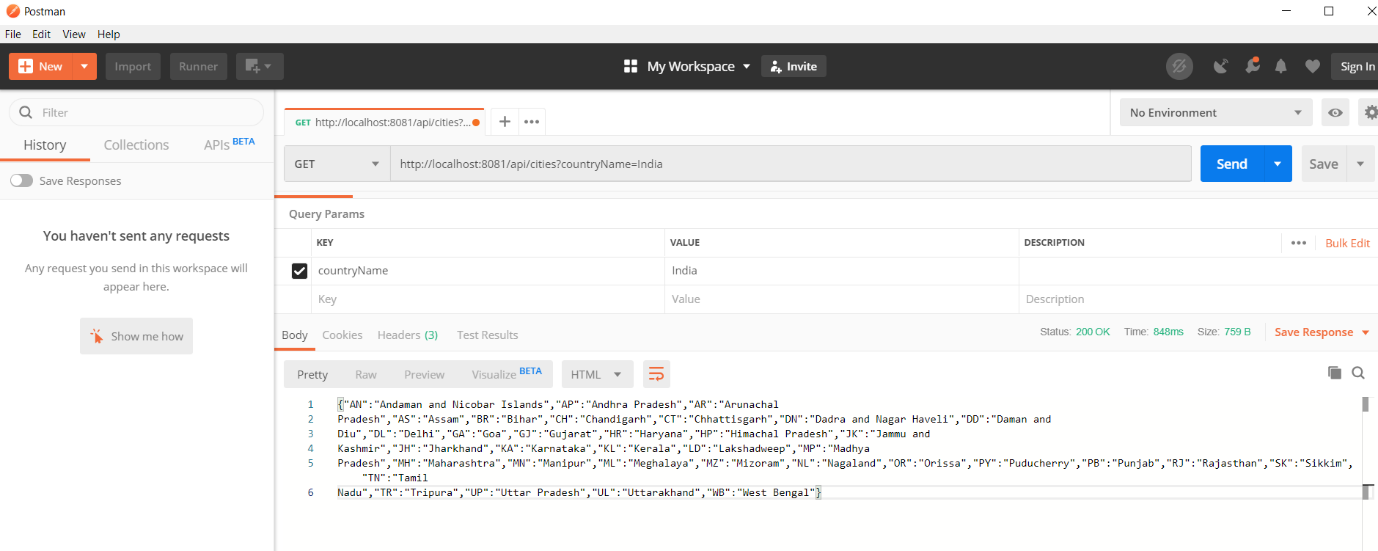
The format of the webservice is as below.

api.openweathermap.org/data/2.5/weather?q=Australia

* [http://www.westclicks.com/webservices/?f=json&c={countryName}](http://www.westclicks.com/webservices/?f=json&c=%7bcountryName%7d)

This webservice expects the country name for which the city details are required.

e.g. City details for India as below as seen below in postman



**Challenge2: API Kit router was not getting generated while importing the Weather API published to Exchange in AnyPoint Studio.**

After designing the weather API in AnyPoint platform design center, the API was published to Exchange to make it discoverable/re-usable. In order to develop the actual API functionality, I tried to import this API from Exchange in AnyPoint Studio while creating the project. However, it was giving some errors in pop up window.

**Solution**:

1. I re-tried to import the API from Exchange with new workspace for the project, the issue still remained.
2. After couple of times of failed attempts of step A, I redesigned the API (i.e. recreated the API specification design) and published back to Exchange. Then, I was able to create API Kit router after successful import of Weather API from Exchange.

**It seemed there was an issue with publishing of Weather API previously.**

# 

# Design Decisions

Since, I encountered an issue with provided online (as well as dockerised) version of SOAP webservice as it was not functional, I tried to research if there is any substitute web service available freely that can be used instead of this non-functional web service as part of exposing couple of operations **GetWeather** and **GetCitiesByCountry.**

I came across below services that can be helpful in this exercise.

1. https://openweathermap.org

This webservice is useful to get all the weather related details for a given country.

This can be called in below format.

api.openweathermap.org/data/2.5/weather?q={countryName} where countryName can be any Country for which the weather details need to be retrieved.

However, this API needs to be subscribed to make use of it. An unique API key is provided as part of its subscription which needs to be passed to this API in the form of header key APPID.

1. [http://www.westclicks.com/webservices](http://www.westclicks.com/webservices/?f=json&c=France)

This webservice is useful to get all the states/cities for a given country.

This can be called in below format.

http://www.westclicks.com/webservices/?f=json&c={countryName} where countryName can be any Country for which the cities details need to be retrieved.

**Note:** For limited testing scope, only below countries are configured in the application for which weather and cities details can be retrieved.

'Australia','India','France','New Zealand','Singapore'

# Exception Handling approach

The API is designed in a way to handle exceptional scenarios while calling it.

The scenarios include below use cases.

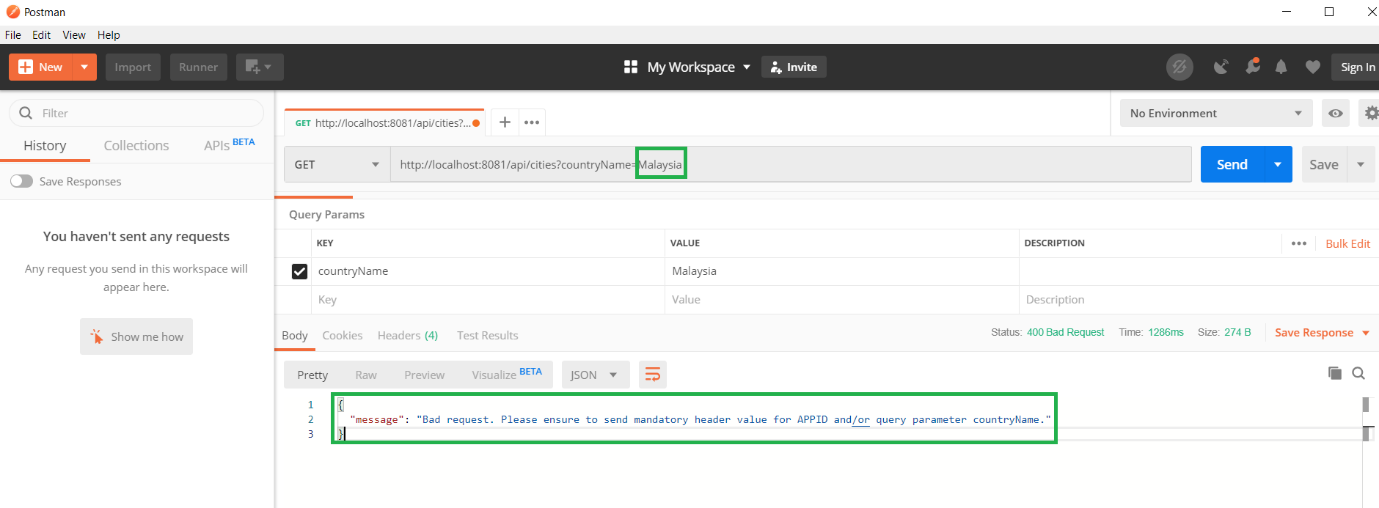
1. **Invalid country selection:**

Although, API has been designed in a such a way that only configured countries can be selected, there has been additional exception logic implemented in the code to check if the country passed in by the API consumer is a valid country. If the country is found to be a valid country, API execution continues. In case of invalid country, a meaningful error message with proper HTTP status code is presented.

e.g. This is how API consumer is forced to select country from the predefined set of countries. i.e. using predefined enum in API design.



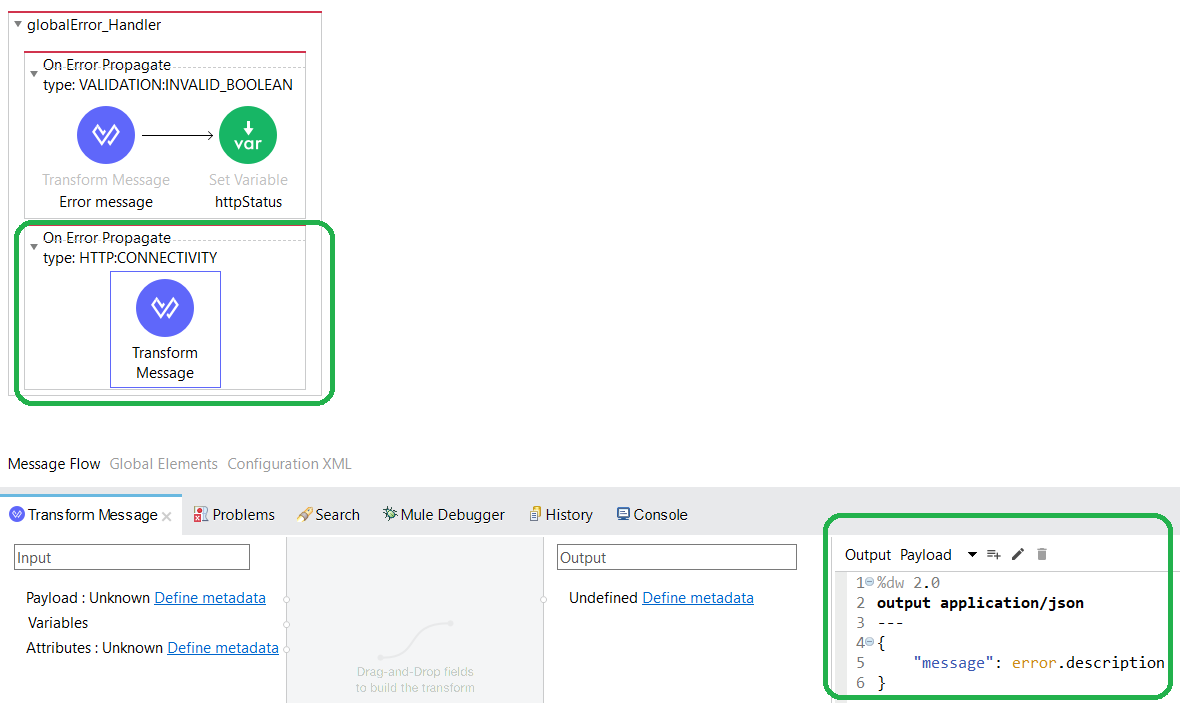
Error message when invalid country is passed to an API. E.g. Malaysia



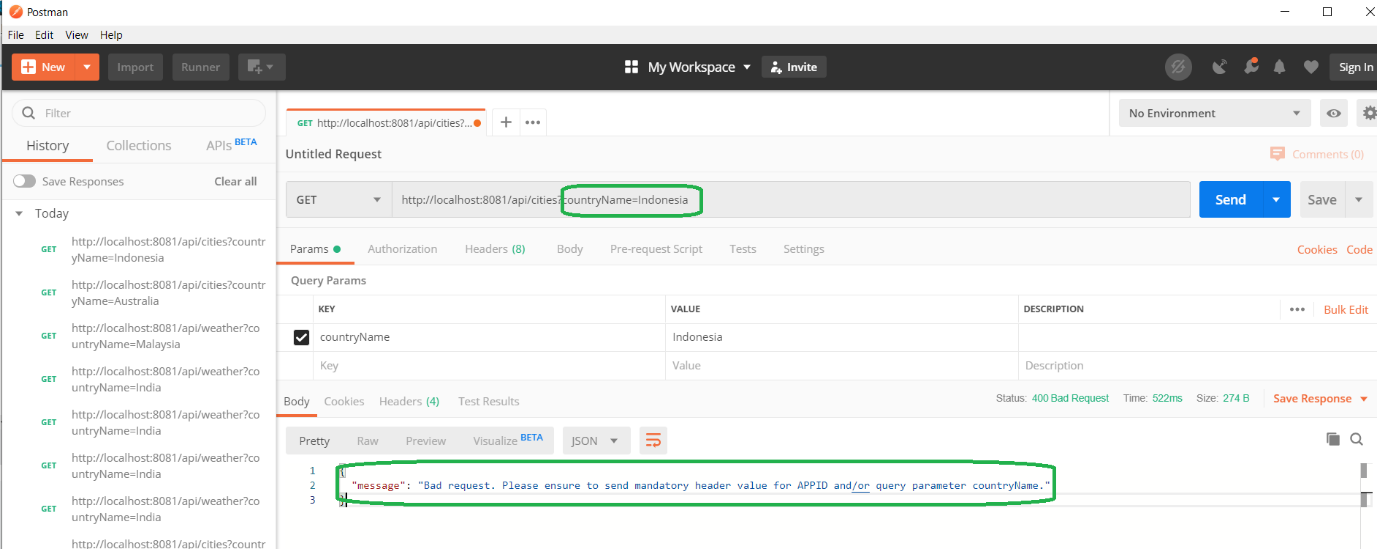
1. **Connectivity issues with the external web service.**

Since as a workaround for an issue with the original GlobalWeather SOAP based webservice, couple of freely available webservices have been used to cater to this requirement to expose the operations defined in GlobalWeather service. Sometimes, these webservices may not be functional such as down for maintenance/server upgrade etc. Hence the webservice consumers need to be notified in such scenarios.

Error handling code below



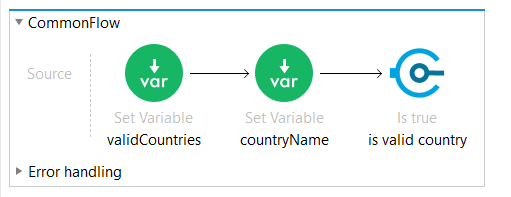
Error presented to the consumer when country is provided other than the valid country in postman below.



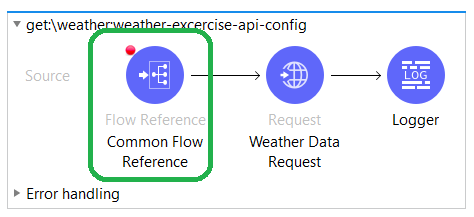
# Modularity and maintainability

1. Since the logic for retrieving countryName and validating it, is common to both the endpoints (i.e. /weather and /cities), it has been extracted in a separate common flow which can be used in both of these flows through a flow reference connector.

Common flow logic as below.

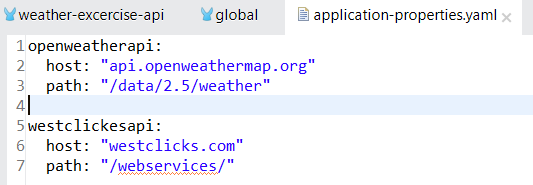


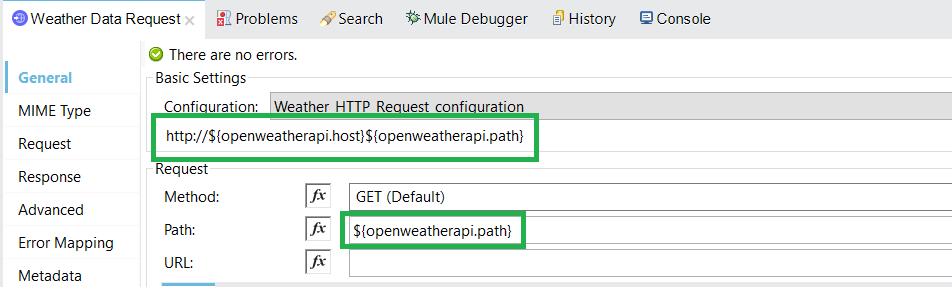
Referencing it in the weather flow as below.



This approach helps use of re-usability and avoid redundant code logic so that making a change at one common place can apply to both of the flows.

1. The external use of REST based webservices have been configured in the application code in such a way that, the change can be incorporated in the external configuration file (i.e. application-properties.yaml) if there is any change in future.





Project structure as below.

