Technical Specification Document for Weather REST API Service

11th Nov, 2019

Version 1

|  |  |  |  |
| --- | --- | --- | --- |
| Revision History | | | |
| DATE | VERSION | DESCRIPTION | AUTHOR |
| Nov 11, 2019 | 1 | Initial version | Tarunpreet Kaur |

# **Introduction**

The aim of this exercise was to wrap the SOAP service provided(Global Weather web ) in a RESTful API. The SOAP web service GlobalWeather has two operations:

1. GetWeather
2. GetCitiesByCountry

The URL of the web service provided <http://www.webservicex.com/globalweather.asmx?WSDL>, was giving a 404 (Not found error), rather I used the Node application provided in the exercise document for this project.

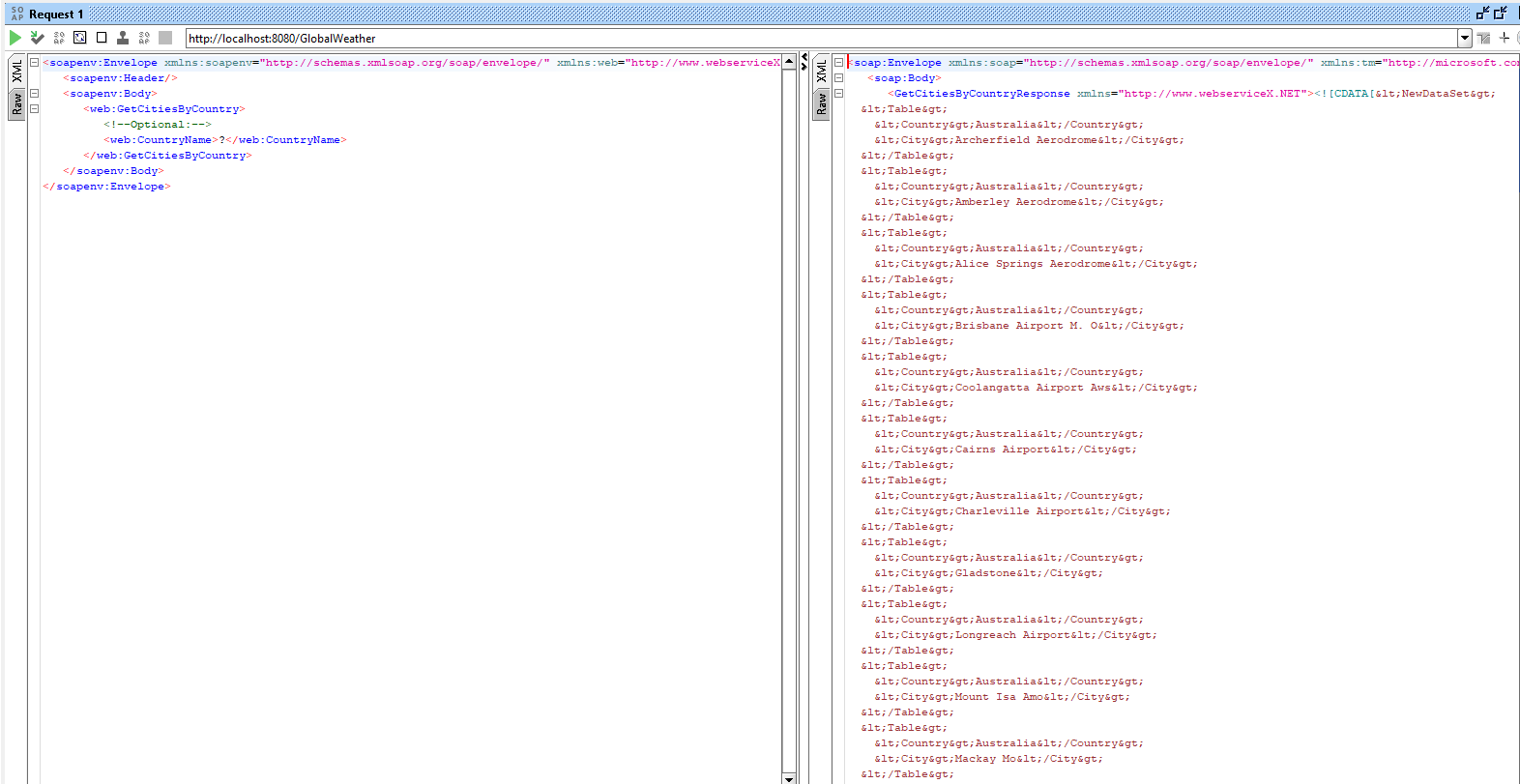
GlobalWeatherProject REST API exposes the operations from the external weather SOAP webservice (<http://www.webservicex.com/globalweather.asmx?WSDL>). This REST API has been developed in Mule 4.

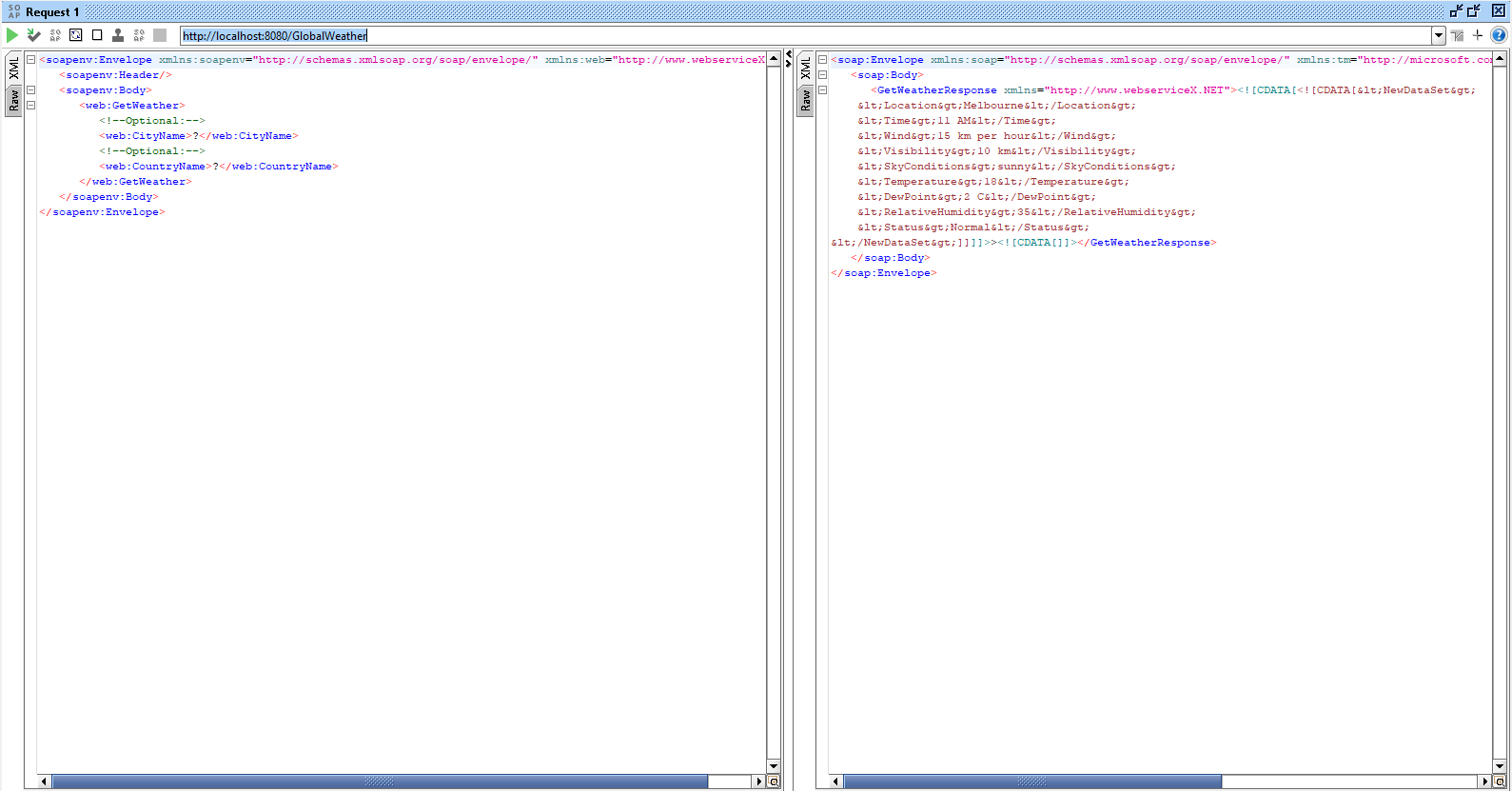
# **Initial steps**

* Provided external webservice URL <http://www.webservicex.com/globalweather.asmx?WSDL> was not available(404 Not Found). So, Mule REST API, references the web service provided by Deloitte, downloaded from the DropBox. I ran the web service by going into the directory and ran the following commands:

1. Npm install – To install all modules listed in the package.json file
2. Node server

Both the operations of the service were accessible through SOAP UI.





Provided NodeJS webservice response for getCities and getWeather operations was not formatted xml and it was coming in as a String in the CDATA element. The String had to be parsed, unescaped and then response was to be converted from XML to JSON.

# **Prerequisites**

1. The GlobalWeather Web service should be running. Node should be installed in your system. Extract the dropbox file and go inside the extracted file till you get the package file. Run the npm install and node server command here. Hit the service in SOAP UI as above Code been developed using http://localhost:8080/GlobalWeather?WSDL
2. Access to Mule Platform Anypoint Studio, Design Center.
3. Node JS, SOAP UI, Postman tool to test the REST and SOAP service. Eclipse to test the Java code functionality (I used Eclipse to run the unescapeXML operation, but it is out of scope for the current version).

# **Design Decisions**

I started by designing the RAML file. It had two operations with the Method GET. I opted for Query parameters instead of Path parameters because in both the operations these were optional parameters, whether we pass those or not, the results should be returned. If those were passed the results would have been filtered as per the parameters passed.

# **REST API operations**

Weather REST API provides the below service and respective operation are listed below.

1. **Cities Service: /api/v1/cities**
   1. GET

*Description:*

This operation gets the list of cities where airports are present. It seems to be a misnomer to me on the SOAP side, as it is getting the list of Airports in that Country. But as this is only a POC, I decided to keep it as cities. We can make changes in future if required. The query parameter is optional. In our case it is retrieving same results whether we pass this parameter or not.

*Query Parameter:*

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Optional | Example and URI |
| country | String | Yes | country=Australia  URI: http://localhost:8081/api/v1/cities?country=Australia |

1. **Weather Service: /api/v1/weather**
   1. GET

*Description:*

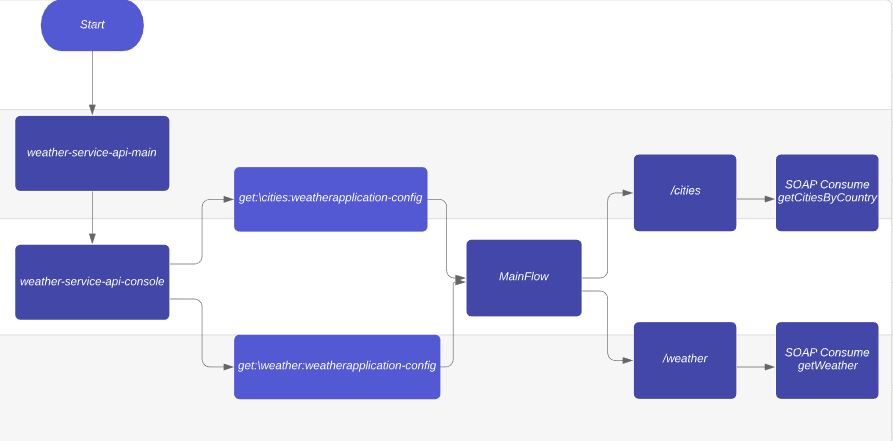
Get the weather data for a given city in the country which will be passed in the query parameters. If the query parameter is not given, API retrieves weather data for the city Melbourne from country Australia. In our case it is retrieving same results whether we pass these parameters or not.

*Query Parameter:*

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Optional | Example and URI |
| country | String | Yes | country=Australia  URI: http://localhost:8081/api/v1/weather?country=Australia&city=Melbourne |
| cityName | String | Yes | city=Melbourne  URI: http://localhost:8081/api/v1/weather?country=Australia&city=Melbourne |

# **Mule flow**

# Themule flow is developed by creating the RAML file in the Designer Center and creating a New Project by selecting API implementation from the Design Center.



## **Weather-Service API Main Flow**

Its an interface for the implementation of the weather service REST API and the it provides the console for the testing the service. Its been built using APIKit plugin in anypoint Studio.

As defined in the RAML file (weather-service-api.raml), based on the url APIRoute in the main flow routes the request to the respective operation in the implemantion flow.

Interface flow is build based on the RAML definition (weather-service-api.raml).

## **Implementation Flow**

Implementation flow been developed in the anypoint studio and its been invoked by the interface weather-service-api-main (API gateway) for the services. Implementation message flow comprises 2 flows for the operations getCitiesByCountry and getWeather from the external SOAP service.

**getCities:**

Its used to consume the external webservice using the queryparameter (countryName) and displays the list of cities from the country. The response is extracted from String format and the escaped characters(&lt;, &gt;) are replaced with equivalent unescaped characters. The String is converted to XML format using Set Payload. This well formed XML is converted to JSON and is set as response. The duplicateKeyAsArray=true is used to change the value to an array as we have duplicate keys in the object.

**getWeather**:

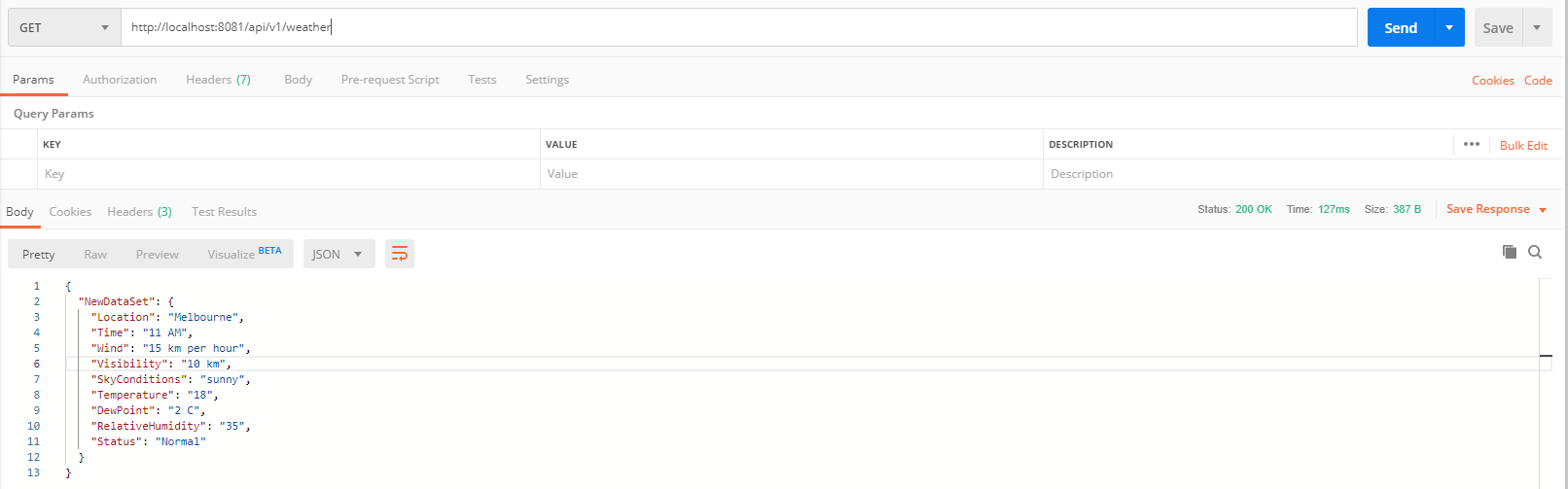
Its used to consume the external webservice using the queryparameters (countryName, cityName) and displays the weather data of cities from the country. The response is extracted from String format and the escaped characters(&lt;, &gt;) are replaced with equivalent unescaped characters. The String is converted to XML format using Set Payload. This well formed XML is converted to JSON and is set as response.

# **API In Action**

Both the operations can be tested in Browser/Postman

* 1. <http://localhost:8081/api/v1/cities>
  2. <http://localhost:8081/api//v1/cities?country=Australia>

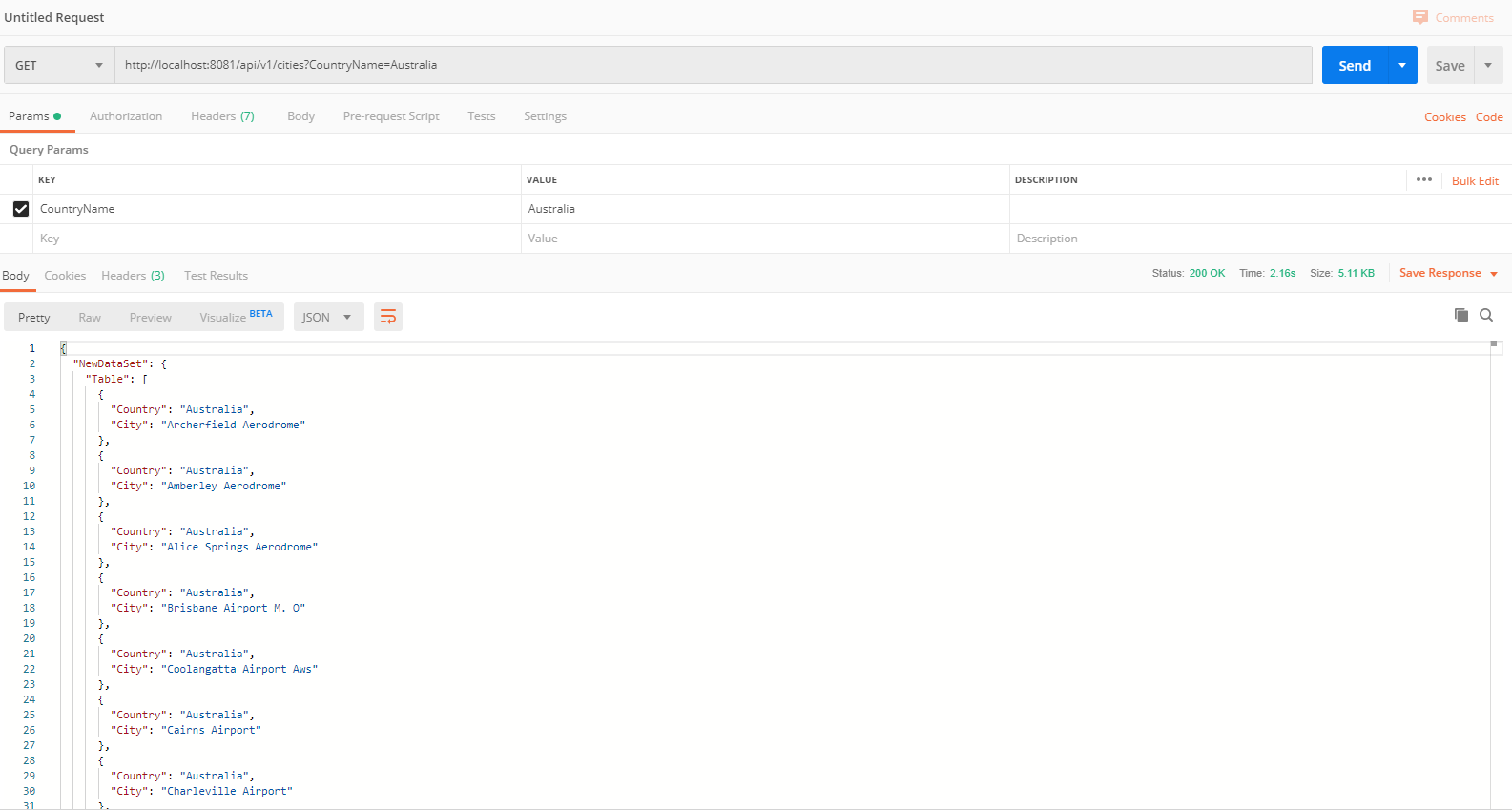
Both the above URLs would return same result.



* 1. <http://localhost:8081/api/v1/cities>

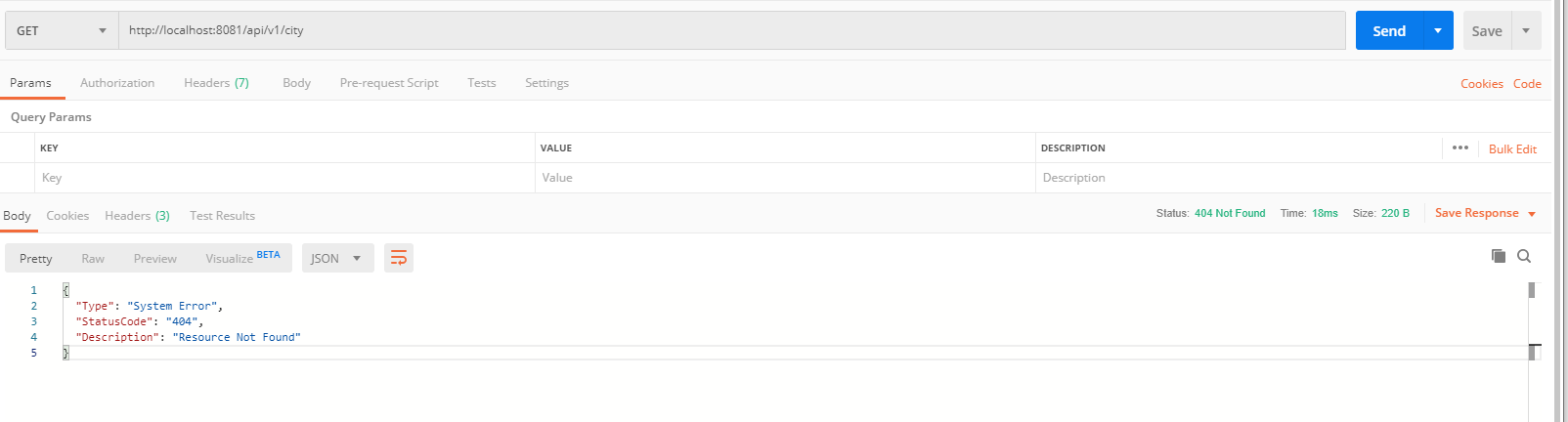
<http://localhost:8081/api/v1/weather?country=Australia&city=Melbourne>

Both the above URLs would return same result.



# **ERROR HANDLING**

I have handled 404, 405, 400 errors in the Error Handling. Please find below the screenshots. The error handling is common to both the operations and we get similar messages.





# **CHALLENGES**

1. The hardest part was extracting the String out of the response and then converting the response to JSON. I found a post by mulesoft to unescape the data using Java function as below:

<https://help.mulesoft.com/s/article/How-to-unescape-XML-message>

I did a small POC to implement it in Eclipse and the String data was formatted but faced issues while implementing it through Mule 4 Invoke function as I had to pass the payload as a Stream to the Java function and then the unescaped was to take place.

Another way which I thought was to use replace and substring function in the data weave to replace the escaped characters with its proper format. It might not be a correct way to use replace and substring but due to time constraints I had to resort to this option.

1. Faced some issues with RAML syntax but it was a good learning curve.

# **SCOPE FOR ENHANCEMENT**

1. Using Java code to pass the payload(Through stream) and unescape the data in Java code and return the payload back to the Mule Flow.
2. Changing the name of the operation(GetCitiesBy Country)as it seems to be a misnomer