**Design Specifications**

**Introduction:**

This document will provide the information regarding design specifications and challenges faced during the implementation of this mule application. Moreover, this document will also provide the brief explanation of each component used in the flow to achieve the expected results.

**Technical Specifications:**

* Firstly, I have configured docker to access the SOAP webservice to check the WSDL and its related operations to retrieve the desired data in response.
* To consume this SOAP web service firstly I have created RAML and related datatypes with Custom error message. The reason behind the implementation of RAML first is basically to follow the contract first approach in the API implementation.
* In this mule application, I have implemented a RAML, Custom error message and two datatypes to design the APIs, please follow below table for more details:

|  |  |  |
| --- | --- | --- |
| **Name of RAML** | **Type** | **Description** |
| WeatherApp\_Del.raml | RAML | It is a main raml file to store the information about designing of all APIs. |
| Country.raml | Datatype | Raml file to store the properties for the country datatype. |
| Weather.raml | Datatype | Raml file to store the properties for the weather datatype. |
| CustomErrorMessage.raml | Datatype | Raml file to store the properties for the country datatype. |

* In the flow which is generated by the main RAMl includes “HTTP listener” as a source component. Then the payload passes over to “APIkit Router” in process pane which route the request according to the URL and provided query params in the input.
* In next step, the respective sub flow is invoked which uses the “Flow reference” component to redirect the request to the assigned subflow.
* In “weatherApp\_SubFlow”, firstly the set variable used to set the value of the variable.
* Then the payload redirects to “Transform message” component where input xml body is prepared to pass to the “Webservice Consume” component. This component sends a request to the provided SOAP url and provide the response to the “Transform Message” component.
* In this component, payload body is extracted and pass over as output XML to “Transform Message” component again.
* In this component, payload is processed as string to remove all unwanted characters and passed it to another component to read the specific data from the payload and map it with specified key for the Json.
* In this flow, I have also implemented “Error propagate” functionality to trigger the error during the execution of flow and halt the flow.
* I have also implemented MUnit test functions, to test the final expected output with the provided set of output parameter.
* The same set of design functionality is used for the implementation of another subflow which retrieves weather related information by providing input country and city name.

Challenges faced in implementation:

* The response received from the SOAP webservice for the operation “getCitiesByCountryName” was in CDATA format and included “&lt;” and “&gt” on the place of “<” and “>”. So to overcome this issue, I have written data weave to first transform the xml message in string and replace the undesired characters and again set the payload as per the required format.
* The response received from the SOAP webservice for the operation “getWeatherByCountryAndCityName” was in CDATA format and included “&amp lt;” and “&amp gt;” on the place of “<” and “>”. So, to overcome this issue, I have again written data weave to first transform the xml message in string and replace the undesired characters and again set the payload as per the required format.