Technical Requirements Specification

for

Global Weather Service

Version 1.0

Prepared by Amit Tarneja

25 August 2019

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 3

1.1 Purpose 3

1.2 Intended Audience and Reading Suggestions 3

1.3 Project Scope 3

1.4 References 3

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Features 3

2.3 User Classes and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Challenges 4

2.6 Setup Instructions 6

3. Sample 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Amit Tarneja | 25th August 25, 2019 | Initial Version | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this document is to explain the technicalities involved in building a REST adapter for a legacy SOAP end-points.

## Intended Audience and Reading Suggestions

The document is indented for all the staff members who will use these new REST endpoints.

## Project Scope

Digital transformation is a challenge that every company is facing nowadays. Customers are demanding new experiences, partners and developers want greater access, and employees want greater convenience and be able to access systems from wherever they are. Business who choose not to take on digital transformation put themselves at risk. The project is an initiative to avoid that risk and provide the business an upper hand to simplify (Apify) their complex integration end-points.

## References

<https://www.mulesoft.com/resources/cloudhub/creating-digital-transformation-roadmap>

<https://programmerspub.com/blog/general/raml-best-practices#Use-RAML-inheritance>

# Overall Description

## Product Perspective

This is the first initiative from the management to go on the digital transformation path. This prototype project refers to an initial roll-out of the system into production, targeting a limited scope of the intended final solution. The pilot will help the management to decide whether they are going to invest in something available through partners or re-invent the wheel by building and maintaining it as part of the product.

## Product Features

The service is developed using the MuleSoft proprietary product (Anypoint Studio IDE). A REST wrapper got created around the SOAP-based service to simplify its usage and consumption.

## User Classes and Characteristics

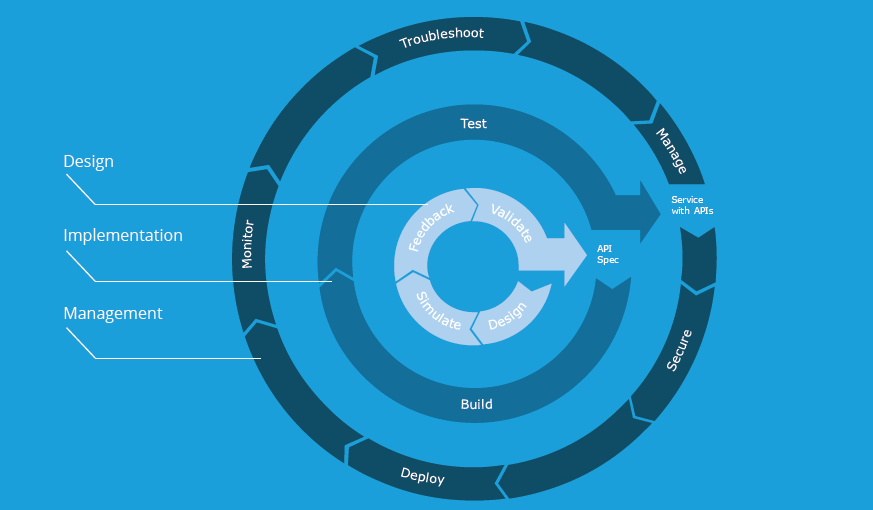
The typical user of the system will be another upstream client that is consuming the existing SOAP-based endpoint. We tried simulating the behavior using URL and the interface that was the outcome of the new deployment using MuleSoft.

## Operating Environment

For this initiative to work, we need to have Node JS and Mule runtime 4.1.4 installed on our machine. I faced some challenges with the existing setup provided by the client, so have documented those in the section below. Please follow the instruction to get the desired outcome from this POC.

## Design and Implementation Challenges

The design of the API is fairly simple, and it caters to most of the best practices followed in the industry whilst designing a new API using RAML.



Some of the best practices followed while designing and implementing the API are as follow:

1. Use Spec Driven Development
2. Think about the API
3. Modularize and Reuse
4. Mock Your API and get User Feedback
5. Resources and Naming
6. HTTP Codes
7. HTTP Verbs
8. Spec driven development

It is really important to define a contract and start coding to contract, and it saves tons of development time.

1. Think about the API

The API is designed for long-term usage. The API is built to scale, and it is backward compatible with the previous version of the SOAP end-points.

1. Modularize and Reuse

I have used a modularized RAML file and also used traits to promote reusability. I have also used modularization in the Mule application so that the common flows can be used across the application, and I can have an optimized code.

1. Mock your API and get user feedback

RAML and Mule designer can be used to create and mocking service so that we can get feedback from the users of this service. This way, we can get early feedback from the users so that we can incorporate the changes upfront in the development cycle.

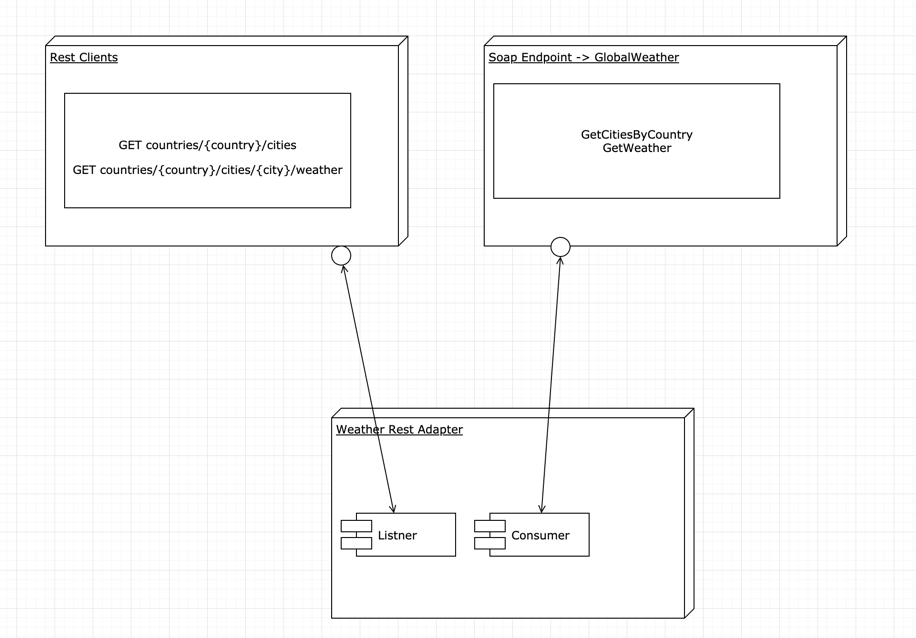
1. Resourcing and Naming

Followed the general REST API guidelines while naming the resources.

1. HTTP codes and verbs

Each resource has at least one HTTP verb and returns the accurate HTTP code for the resource-method pair.

The below figure shows different architecture components involved and their interactions.



The Adapter was built/deployed using Mule EE 4.1.4 (reason explained below), and it listens for the incoming requests on 8081 port on the localhost. The listener then fetches the required parameters from the URI params and generate an XML to call the SOAP WS running on port 8080 on your local machine. With the help of various transformers (utilizing the strength of Dataweave 2.0), the SOAP result is transformed into a human-readable form (JSON in this case).

Challenges:

1. GetWeather response is embedded in a CDATA elements, and that caused a failure in one of the transformers that expect the payload as XML. To resolve the issue, I had to modify the GetWeather response placed in the mock folder. For getting the desired result, please use the modified image of the docker files attached with the GitHub code.
2. I had to install the 4.1.4 version of the Mule execution environment as I got “cannot start embedded container munit” error while running the Munit test cases. Looks like it is a known issue with 7.3.4 Anypoint studio embedded with Maven.

CAUSE

This error is observed in Anypoint Studio 7.3.3 and 7.3.4. The issues occur on projects which use Studio's embedded Maven for its project.

SOLUTION

The issue in Anypoint Studio is released in Anypoint Studio 7.3.5. To resolve the issue, you have to upgrade to Anypoint Studio 7.3.5.

## Setup Instructions

1. Install Anypoint studio with Mule EE 4.2.0
2. Install the Runtime environment as 4.1.4. For more details, please refer to the above explanation.
3. Install NPM on your machine, if not already installed.
4. Clone or download the code from the GitHub repository <https://github.com/amittarneja/weather-rest-adapter>.
5. Go to the folder weatherExcerciseDockerFile and run the npm install command.
6. If you get the message to fix the vulnerabilities, then run npm audit fix to fix the issues.
7. Type the command node server.js once all the vulnerabilities are fixed.
8. Launch http://localhost:8080/GlobalWeather?WSDL and check if you see the content of the WSDL file.
9. Import the weather-rest-adapter in Anypoint studio as an Anypoint studio project.
10. Once imported, Right-click on the project name and click Run As->Mule Application
11. Once the application is deployed, you should be able to access the below endpoints.

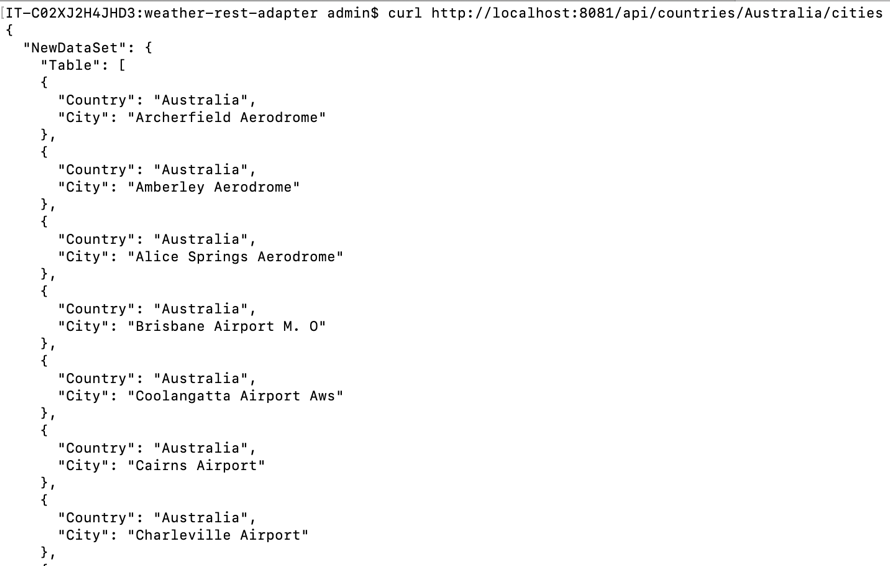
- /countries/{country}/cities ex. http://localhost:8081/api/countries/Australia/cities

- /countries/{country}/cities/{city}/weather ex. http://localhost:8081/api/countries/Australia/cities/Melbourne/weather

# Sample

One of the benefits of using REST is that it is easy to test. There are several ways of testing a REST application, and I have used CURL and the app deployed by Mule to validate the new API’s.

1. Open the terminal app on Mac or Chrome on windows to hit the URL. curl <http://localhost:8081/api/countries/Australia/cities> or simply paste the URL in the address bar of the browser.



1. Use the below endpoint for the second API <http://localhost:8081/api/countries/Australia/cities/Melbourne/weather>



1. We will get Resource not found error in the case where any parameter is missing from the request.



1. I also implemented some checks to validate the input from the user. Currently, the error message in only dumped into the logs and not propagated in the browser. Use the below command to simulate that.

curl http://localhost:8081/api/countries/Newzealand/cities

