About the document

The purpose of the document is to give more details about the API design, design decisions taken during the design and development of the Weather Service in Mule Platform.

Weather Service is a REST Service deployed in the Mule container which acts as a proxy to the SOAP-Based Service - http://www.webservicex.com/globalweather.asmx?op=GetWeather.

REST API Design

RESTful API Modeling Language (RAML) is used to define the interface of the Weather Service. It uses JSON as the messaging format.

baseResponse.json schema defines the generic request attributes. It is included in the response schema of the API.

Input: HTTP GET - http://host:port/api/weather/city/{cityName}/country/{countryName}

Where cityName and countryName are mandatory.

Output:

```
{
   "responseCode": "success",
   "responseSubCode": "data_returned",
   "responseMessage": "weather data returned successfully!",
   "responseTimestamp": "2016-07-20T12:37:04.000-0600",
   "location": "Tamworth Airport, Australia (YSTW) 31-05S 150-50E",
   "wind": "from the NNW (330 degrees) at 10 MPH (9 KT):0",
   "visibility": "greater than 7 mile(s):0",
   "skyConditions": "mostly cloudy",
   "temperature": "66 F (19 C)",
   "dewPoint": "59 F (15 C)",
   "relativeHumidity": "77%",
   "pressure": "29.80 in. Hg (1009 hPa)"
}
```

The output of the REST Service gives the weather details and the status of the request whether is successful or there were errors.

The REST Service returns errors for various error conditions.

- 404 - Data Not Found

- 424 - Target System Error
- 500 - Internal Server Error

Please refer to the RAML file for more details about the REST API

__Environment__

The Application requires:

- Mule 3.8 EE
- Java 8

__About the code__

The code is developed using the Mule 'APIKit' in Anypoint platform. APIKit promotes API based development approach. APIKit generates the code based on the API definition for both SOAP-based or REST-based APIs.

```
  <flow name="weather-service-main">
        <http:listener config-ref="weather-service-httpListenerConfig" path="/api/*"
doc:name="HTTP"/>
        <apikit:router config-ref="weather-service-config" doc:name="APIkit Router"/>
        <exception-strategy ref="weather-service-apiKitGlobalExceptionMapping"
doc:name="Reference Exception Strategy"/>
    </flow>
```

APIKit generates the HTTP listener for the API, creates flows for each method and defines exception strategies.

Weather Service REST API contains only one GET method in the RAML, hence the following flow is generated for the method.

```
<flow name="get:/weather/city/{cityName}/country/{countryName}:weather-service-config">
```

The flow is as follows:

- Calls the subflow 'WeatherServiceSoapFlow'
    - WeatherServiceSoapFlow sub flow maps the URI parameters to the GetWeather request. All mappings uses dataweave transformation.
    - WeatherServiceSoapFlow calls CallWeatherServiceSoapFlow which "ws:consumer" to create the SOAP Request and sent it to the GetWeather SOAP Service.

- Calls the subflow ValidateWeatherServiceSoapFlow to validate the response. The response is checked for data not found message and error status.
    - If data not found, org.mule.module.apikit.exception.NotFoundException is thrown which will be later mapped to 404 code based on the RAML specification.
    - If error, com.mayvel.common.exception.TargetSystemException is thrown which will be later mapped to 424 code based on the RAML specification.
    - Apikit:mapping-exception-strategy maps the exceptions with the HTTP response codes and allows to set the payload.
        - If org.mule.module.apikit.exception.NotFoundException, 404 response code is set and the response is sent as per the specification.
        - If com.mayvel.common.exception.TargetSystemException, 424 response code is set and the response is sent as per the specification.
        - All other exception s are mapped to 500 response code.

Unit Testing

The application uses MUnit for the integration testing. The external responses are stubbed in the test/resources/stubs.

The test cases tests the success and all error scenarios.

Note: There is an issue with the negative test cases which is causing the test cases to fail. It might specific to the versions of Mule, MUnit and APIKit. More analysis is required.

Testing

The REST Service can be tested through the API Console provided by Mule or through any other REST Client.

GET URL: http://localhost:8081/api/weather/city/Tamworth%20Airport/country/Australia