Using REST with a Proxy Service

If you have a REST front-end client, REST back-end service, or both a REST client and service, you can use a proxy service in the ESB to handle the communication between the front end and back end. This page describes the following scenarios:

- REST Client and SOAP Service
- SOAP Client and REST Service
- REST Client and REST Service
- JMS Client and REST Service

REST Client and SOAP Service

This usecase represents the scenario where a REST front-end client has to communicate in plain old XML (POX) with a SOAP back-end service. This can be easily achieved by placing the WSO2 ESB in the middle with a proxy service.

Assume the front-end REST client is using http/https as its transport protocol, and the back-end service is expecting SOAP1.1. The proxy service configuration would look like this:

```xml
<definitions xmlns="http://ws.apache.org/ns/synapse">
  <proxy name="StockQuoteProxy" transports="http https" startOnLoad="true">
    <target>
      <endpoint>
        <address uri="http://localhost:9000/services/SimpleStockQuoteService" format="soap11"/>
      </endpoint>
      <outSequence>
        <send/>
      </outSequence>
      <inSequence>
        <header name="Action" value="urn:getQuote"/>
      </inSequence>
    </target>
  </proxy>
</definitions>
```

In this scenario, the client sends a REST message to the ESB, which transforms the message to a SOAP1.1 message and sends it to the back-end service. Once the back-end response is received, the ESB sends the response back to the client as a REST message.
To implement this scenario:

1. Start the ESB and change the configuration as shown above.
2. Deploy the back-end service 'SimpleStockQuoteService' and start the Axis2 server using the instructions in Starting Sample Back-End Services.
   You will now send a message to the back-end service through the ESB using the sample Stock Quote Client. This client can run in several modes.
3. Run the following ant command from the <ESB_HOME>/samples/axis2Client directory to trigger a sample message to the back-end service:
   ```
   ant stockquote -Daddurl=http://localhost:8280/services/StockQuoteProxy -Drest=true
   ```

If the message is mediated successfully, it will display an output on the Axis2 server’s start-up console along with an output message on the client console. If you want to see exactly how the REST messages are transformed to SOAP messages, you could place an Apache tcpmon application between the client and the ESB and another between the ESB and the back-end service.

For another example of POX to SOAP conversion, see Sample 50: POX to SOAP Conversion.

**SOAP Client and REST Service**

This usecase represents the scenario where a SOAP front-end client has to communicate with a REST back-end service. This can be easily achieved by placing the WSO2 ESB in the middle with a proxy service.

The proxy service configuration would look like this:
<definitions xmlns="http://ws.apache.org/ns/synapse">
  <proxy name="CustomerServiceProxy"
    transports="http https"
    startOnLoad="true">
    <target>
      <inSequence>
        <filter xpath="/getCustomer">
          <property name="REST_URL_POSTFIX" expression="/getCustomer/id" scope="axis2" type="STRING"/>
          <property name="HTTP_METHOD" value="GET" scope="axis2" type="STRING"/>
        </filter>
        <header name="Accept" scope="transport" value="application/xml"/>
        <send>
          <endpoint>
            <address uri="http://localhost:9764/jaxrs_basic/services/customers/customerservice/customers" format="pox"/>
          </endpoint>
        </send>
      </inSequence>
      <outSequence>
        <send/>
      </outSequence>
    </target>
  </proxy>
  <sequence name="fault">
    <log level="full">
      <property name="MESSAGE" value="Executing default &quot;fault&quot; sequence"/>
      <property name="ERROR_CODE" expression="get-property('ERROR_CODE')"/>
      <property name="ERROR_MESSAGE" expression="get-property('ERROR_MESSAGE')"/>
    </log>
    <drop/>
  </sequence>
  <sequence name="main">
    <log/>
    <drop/>
  </sequence>
</definitions>

In this scenario, the client sends a SOAP message to the ESB, which transforms it to a REST message and sends it to the back-end service. Once the back-end response is received, the ESB sends it back to the client as a SOAP message. The filter mediator is used to identify the required service. Though this example only has one filter, there could be many filters in the real scenario. The REST_URL_POSTFIX property is a well-defined property, and its value is appended to the target URL when sending messages out in a RESTful manner. Similar to REST_URL_POSTFIX, the HTTP_METHOD property is also well-defined and sets the HTTP verb to GET.
To implement this scenario:

1. Start the ESB and change the configuration as shown above.
2. Download, install, and start WSO2 Application Server. The application server ships with JAX-RS, which contains a set of pure RESTful services, so we will use this as our back-end service.
3. Send a message to the back-end service through the ESB using SoapUI. The service used in this scenario is the getCustomer service, which requires the customer ID to complete its task. A sample SOAP message that is used to achieve this is as follows:

   ```xml
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
     <soapenv:Header/>
     <soapenv:Body>
       <getCustomer>
         <id>123</id>
       </getCustomer>
     </soapenv:Body>
   </soapenv:Envelope>
   ```

   This message simply represents the relevant method to be invoked and its associated value. Upon successful execution, SoapUI should display the following message in response.

   ```xml
   <Customer>
     <id>123</id>
     <name>John</name>
   </Customer>
   ```

   For another example, see Sample 152: Switching Transports and Message Format from SOAP to REST POX.

**REST Client and REST Service**

This usecase represents the scenario where a REST front-end client has to communicate with a REST back-end service. Even though they are both using the same architecture, let's assume the front-end client is using JSON whereas the back-end service is using plain old XML (POX). This can be easily achieved by placing the WSO2 ESB in the middle with a proxy service.

The proxy service configuration is as follows:
In this scenario, the ESB simply changes the message type of the client into XML and then passes it to the REST service. Once the ESB has received the XML message, it transforms it back into a JSON message and sends it to the client.
To implement this scenario:

1. Start the ESB and change the configuration as shown above.
2. Download, install, and start WSO2 Application Server. The application server ships with JAX-RS, which contains a set of pure RESTful services, so we will use this as our back-end service.
3. Go to the command prompt and issue the following curl command to simulate the request of a real REST client:

   ```
   ```

Following is the expected output that should appear on the console upon successful execution of the scenario:

```json
{
   "Customer":{
      "id":123,
      "name":"John"
   }
}
```
To implement this scenario:

1. Start ActiveMQ and configure the JMS transport in ESB to work with ActiveMQ.
2. Configure and deploy the JAX-RS Basics sample by following the instructions for building and running the sample.
3. Since we are going to run both WSO2 ESB and the WSO2 Application Server on the same machine, offset the ports of the application server by changing the offset value to 1 in `<AS_HOME>/repository/conf/carbon.xml`.
4. Goto `<ESB_HOME>/samples/axis2Client` and execute the following command:
   ```
   ant jmsclient -Djms_type=text -Djms_dest=dynamicQueues/JmsToRestProxy -Djms_payload="<Customer><name>WSO2</name></Customer>"
   ```
   This sends `<Customer><name>WSO2</name></Customer>` as the payload.

In the application server console, you will see the following line printed:

```
----invoking addCustomer, Customer name is: WSO2
```