Exposing the web services as OData using Integration Gateway of SMP 3.0 SP04

Applies to:

SMP 3.0 SP04

Summary

This document explains about exposing the web services as OData services using the Integration Gateway component of SMP 3.0 SP04.This document covers how developer can write custom coding using Java script in Integration Gateway to send the payload structure to the web service.

Author: Anilkumar Vippagunta

Company: SAP Labs Bangalore

Created on: 24th september 2014

Author Bio

Anilkumar Vippagunta

SAP Mobility Design Center

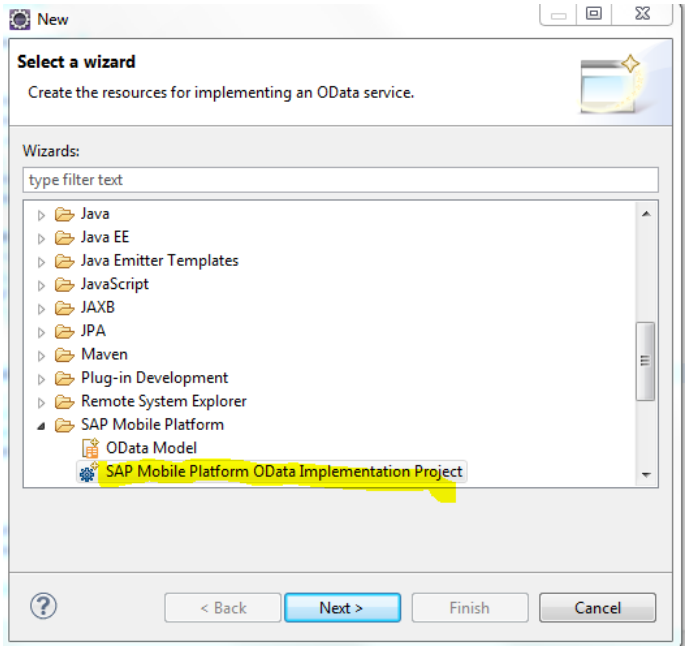
Table of Contents

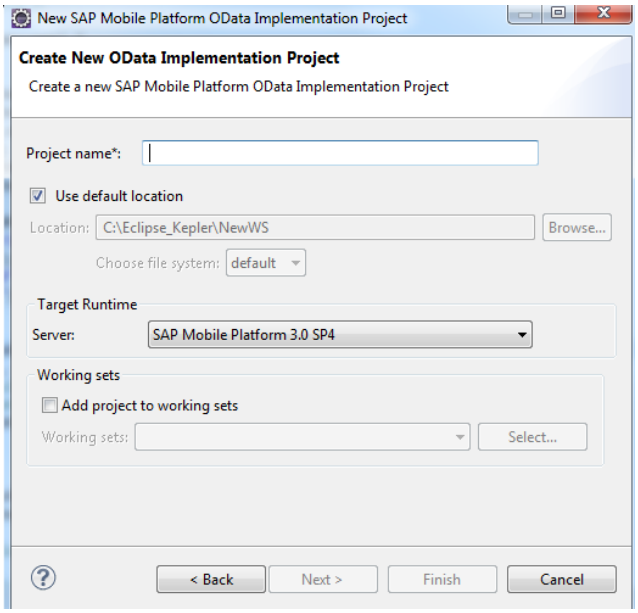
Creating OData Model 3
Creating Operations 5
Define Request and Response Mappings 8
  Request Mapping 8
  Response Mapping 9
Generate & Deploy 10
Testing the Web Services 11
  Web service testing by SOAP UI 11
  Testing the OData Service 12
Sample Code 13
Copyright 16

Creating OData Model

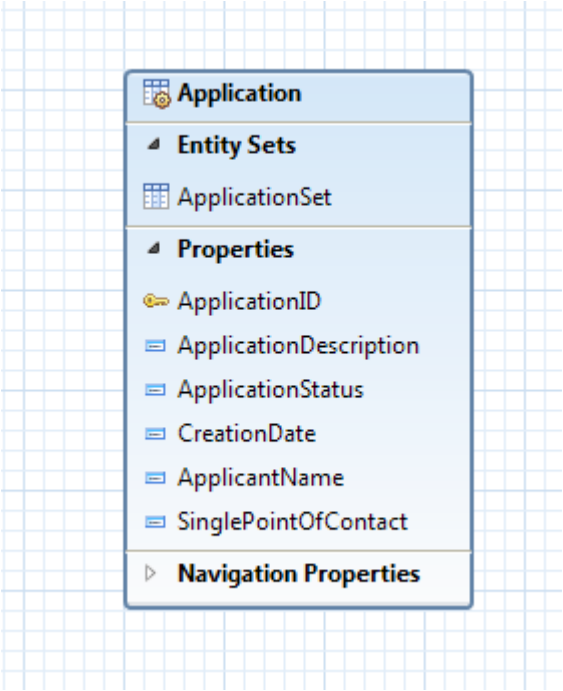
Step1: First choose the web service that you want to expose as a OData Service

Step2: Create OData Model





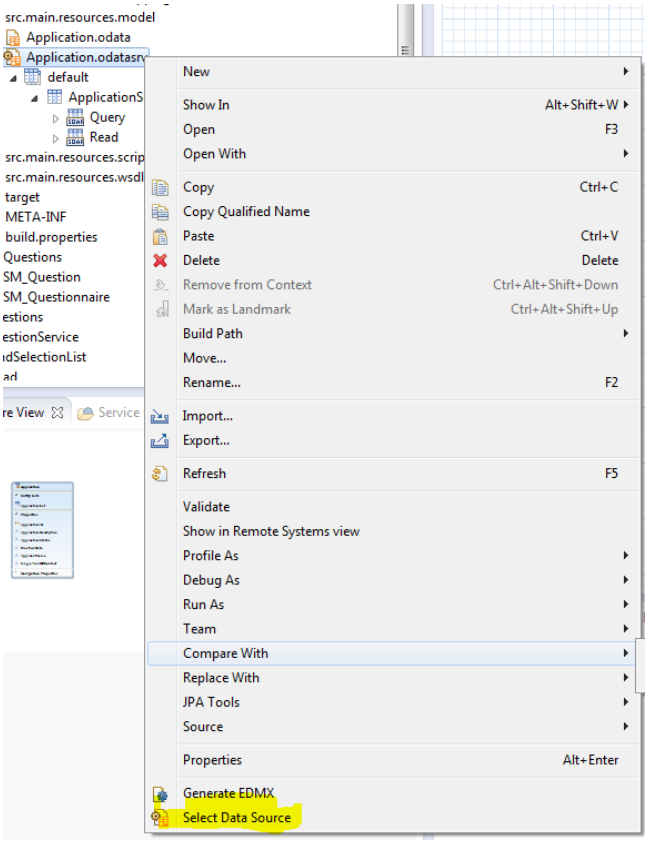
Create OData Model

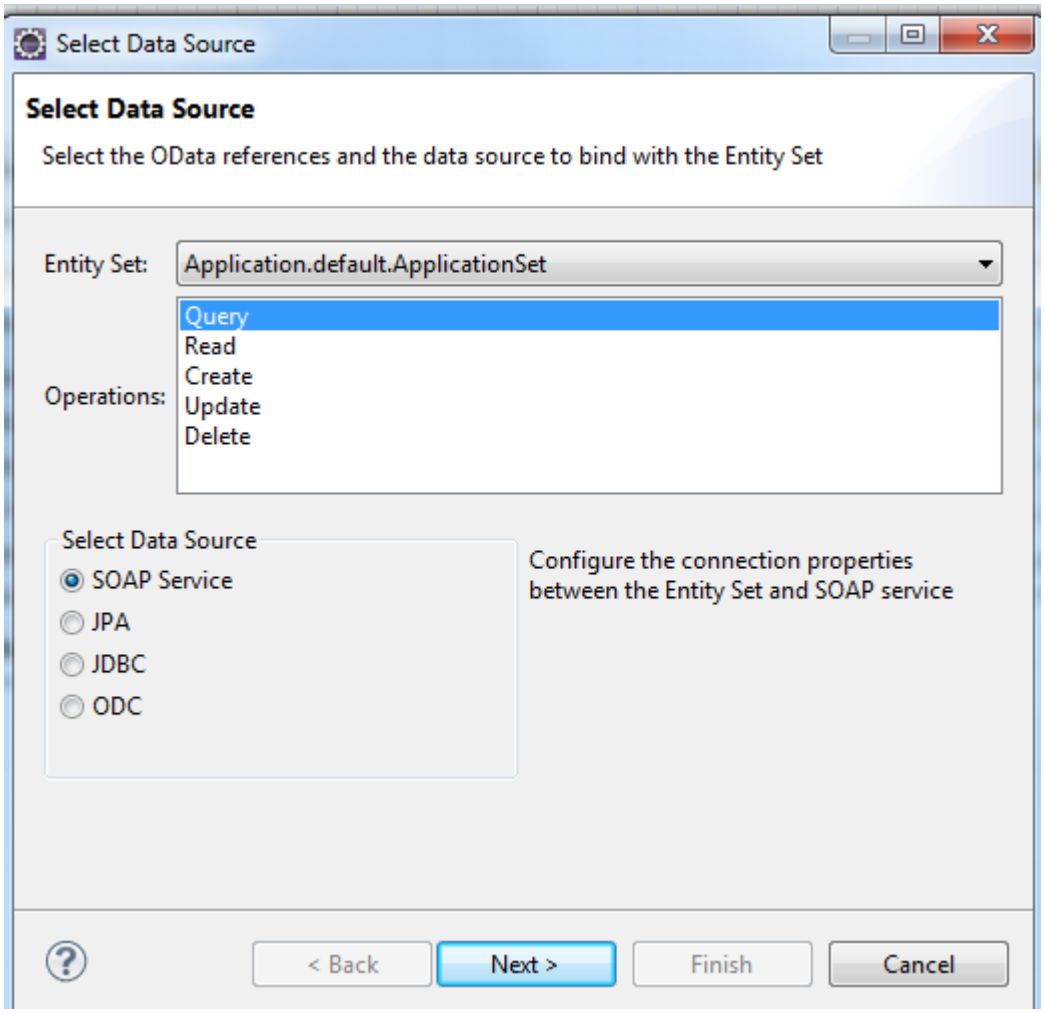

Application
Entity Sets
ApplicationSet
Properties
ApplicationID
ApplicationDescription
ApplicationStatus
CreationDate
ApplicantName
SinglePointOfContact
Navigation Properties

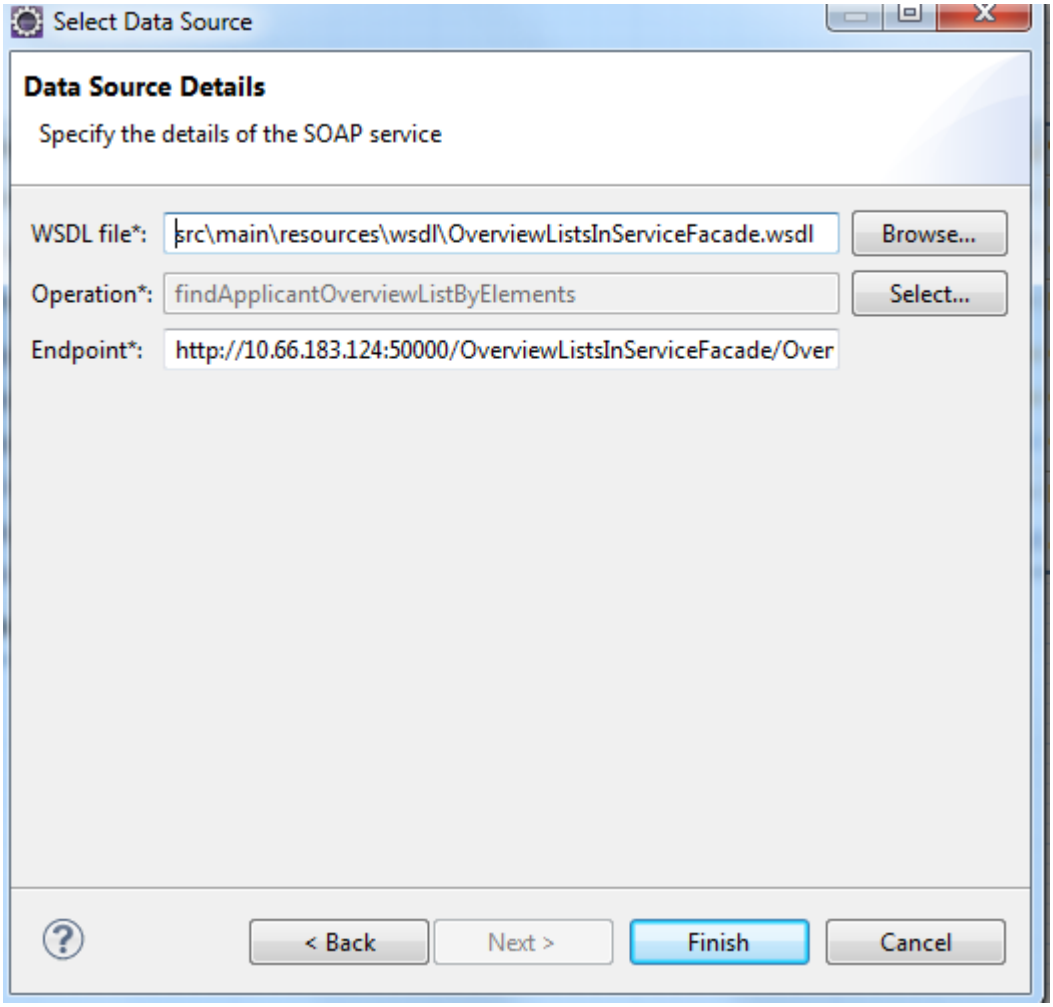
Creating Operations

Step3: Now we can define different operations on the OData model like Create, Query, Read, and Delete.

Right click on the <<Model Name>.odatasrv and choose “Select Data Source”




Select Data Source
Select Data Source
Select the OData references and the data source to bind with the Entity Set
Entity Set:
Application.default.ApplicationSet
Operations:
Query
Read
Create
Update
Delete
Select Data Source
SOAP Service
JPA
JDBC
ODC
Configure the connection properties between the Entity Set and SOAP service
< Back
Next >
Finish
Cancel


Select Data Source
Data Source Details
Specify the details of the SOAP service
WSDL file*:
src\main\resources\wsdl\OverviewListsInServiceFacade.wsdl
Browse...
Operation*:
findApplicantOverviewListByElements
Select...
Endpoint*:
http://10.66.183.124:50000/OverviewListsInServiceFacade/Over
< Back
Next >
Finish
Cancel

Define Request and Response Mappings

Request Mapping

Developer needs to define the request mapping to specify what parameters that we need to send it to the web service interface

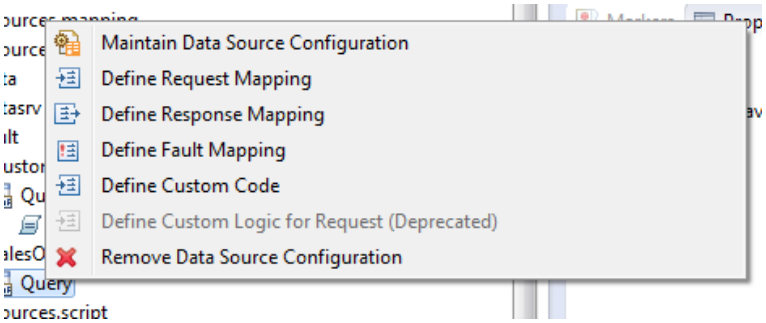
For the above payload (from web service test screen), I've created the below request mapping/custom code .Complete code can be taken from the section Sample Code

Integration gateway (IGW) of SMP SP04 supports writing the custom request mapping coding using Java script and Groovy.

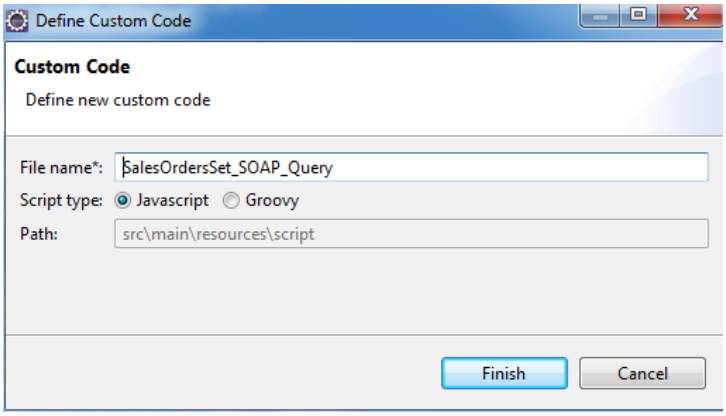
Below code snippet explains

- How we can write custom coding using Java script.
- How to send the authentication details if the web service needs userid/pwd as part of the request (currently hardcoded !)
- How to set the body to the WS request

Right click on the query and select Define Request Mapping.



Now we can choose the script type as SP4 supports writing the customer code in Java script and Groovy

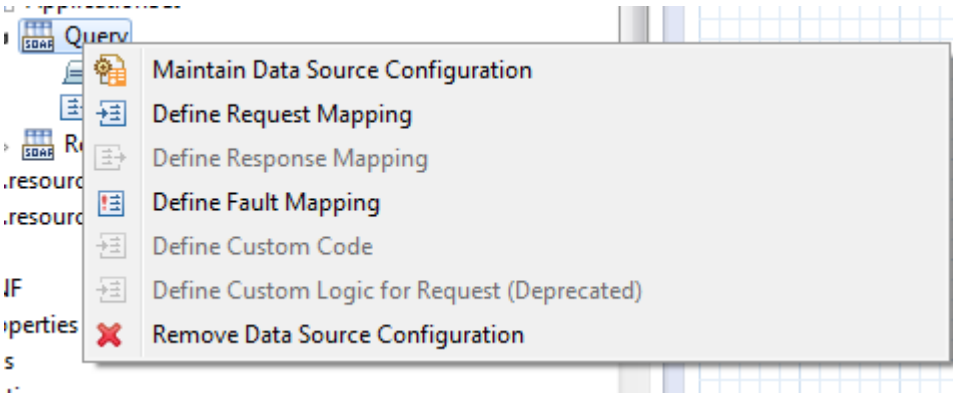


We can see the generated code template like below where we can start sending the payload to the web service

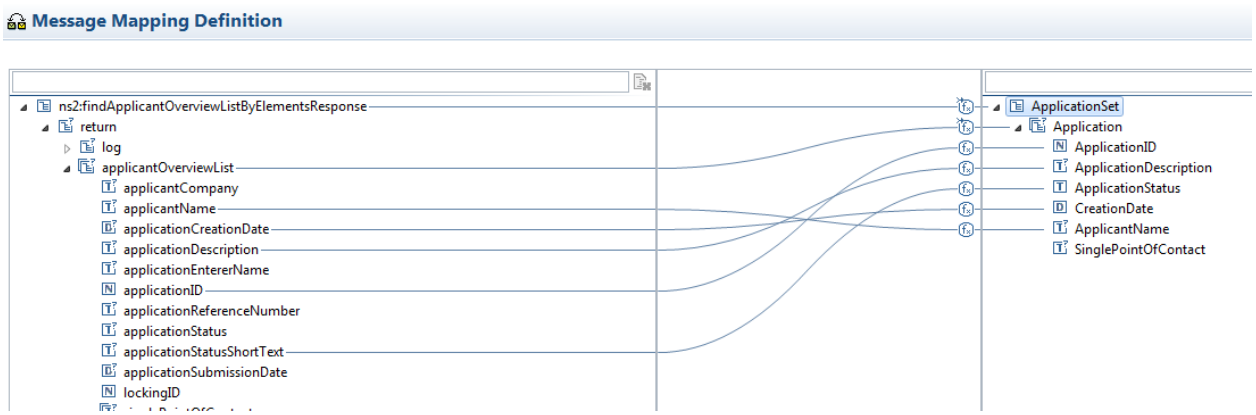
```
function processRequestData(message) {

    importPackage(com.sap.gateway.ip.core.customdev.logging);
    importPackage(com.sap.gateway.ip.core.customdev.util);
    importPackage(org.apache.olingo.odata2.api.uri);
    importPackage(java.util);
    importPackage(com.sap.gateway.core.ip.component.commons);
    importPackage(com.sap.gateway.ip.core.customdev.api);

    var response = new LinkedHashMap();
    message.setHeader("Authorization", "Basic YXBwMDE6YWJjZDEyMzQ=");
    map = new LinkedHashMap();
    map.put("MaxResults", "10");
    map.put("adjustDates", "");


    partneHashMap = new LinkedHashMap();
    partneHashMap.put("arg0", map);
    message.setBody(partneHashMap);
```

1) My web service has authentication.Curently I've hardcoded the encoded value of the userid/pwd
2) I've also sending the same request with tags <arg0><MaxResults/></adjustDates/></arg0> as we see in the SOAP UI Tool.

Response Mapping

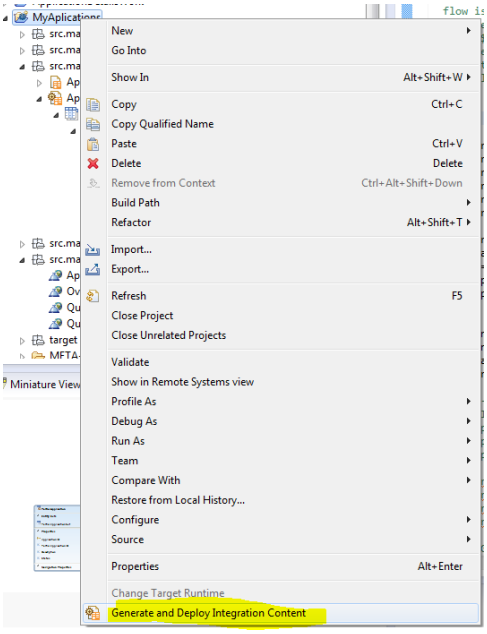


We can map the WS attributes to our OData Model entity attributes through Response Mapping.



Generate & Deploy

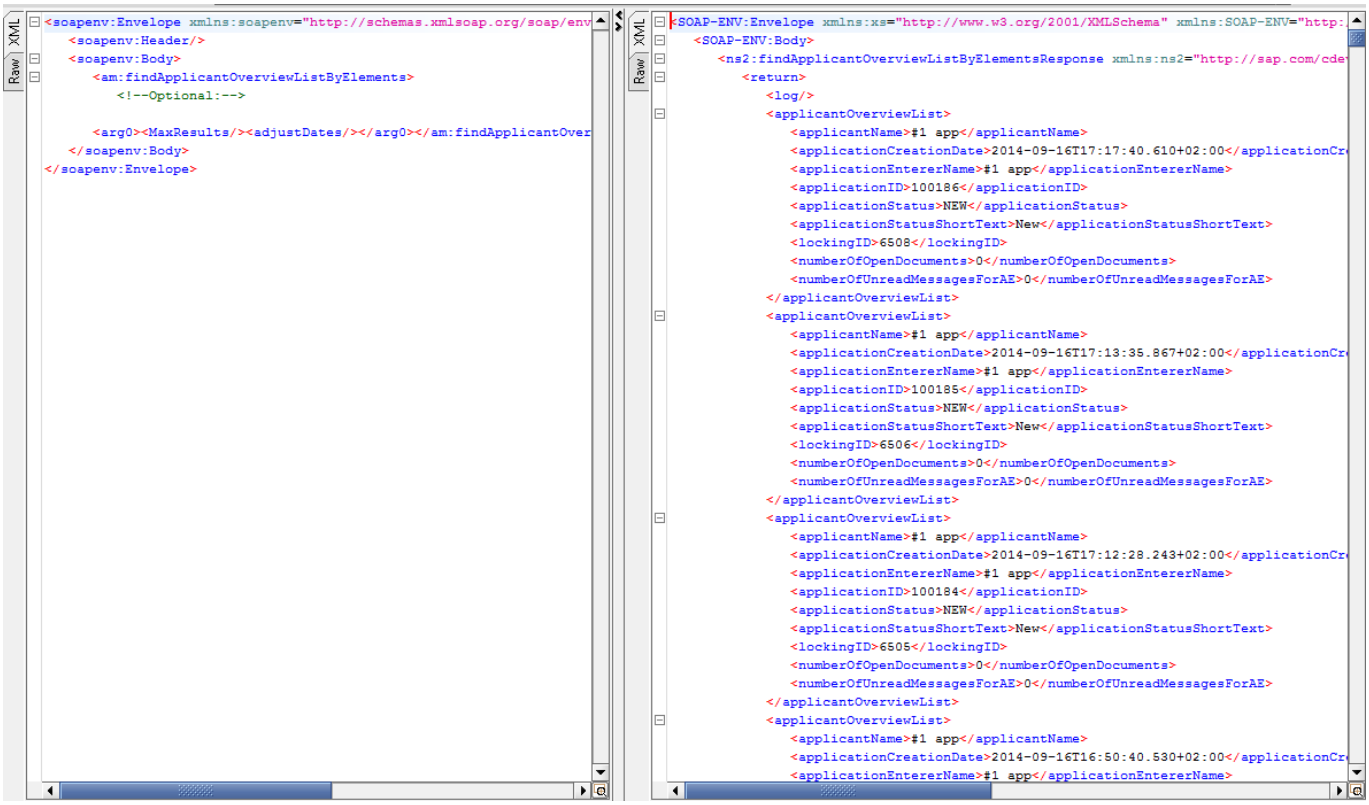
Right click on the project and select “Generate and deploy Integration content”.This will deploy the OData model to the SMP server.



Testing the Web Services

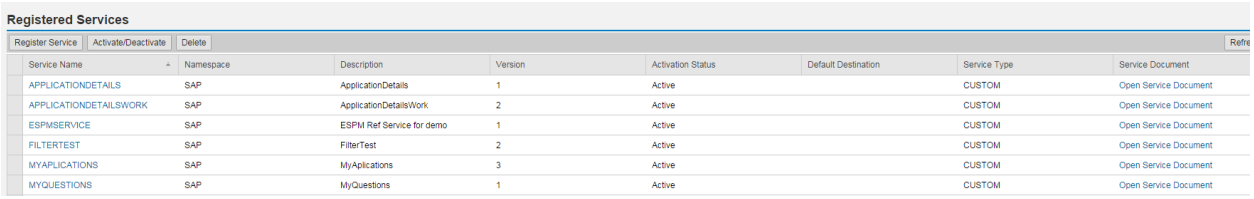
Web service testing by SOAP UI

First test the web service using the SOAP UI tool. We need to create a same payload during our request mapping in Integration gateway.



Testing the OData Service

Launch the SMP Gateway Cockpit https://<<Host>>:<<port>>/gateway/cockpit and we should see the Gateway model that we deployed.



Registered Services

Register Service | Activate/Deactivate | Delete

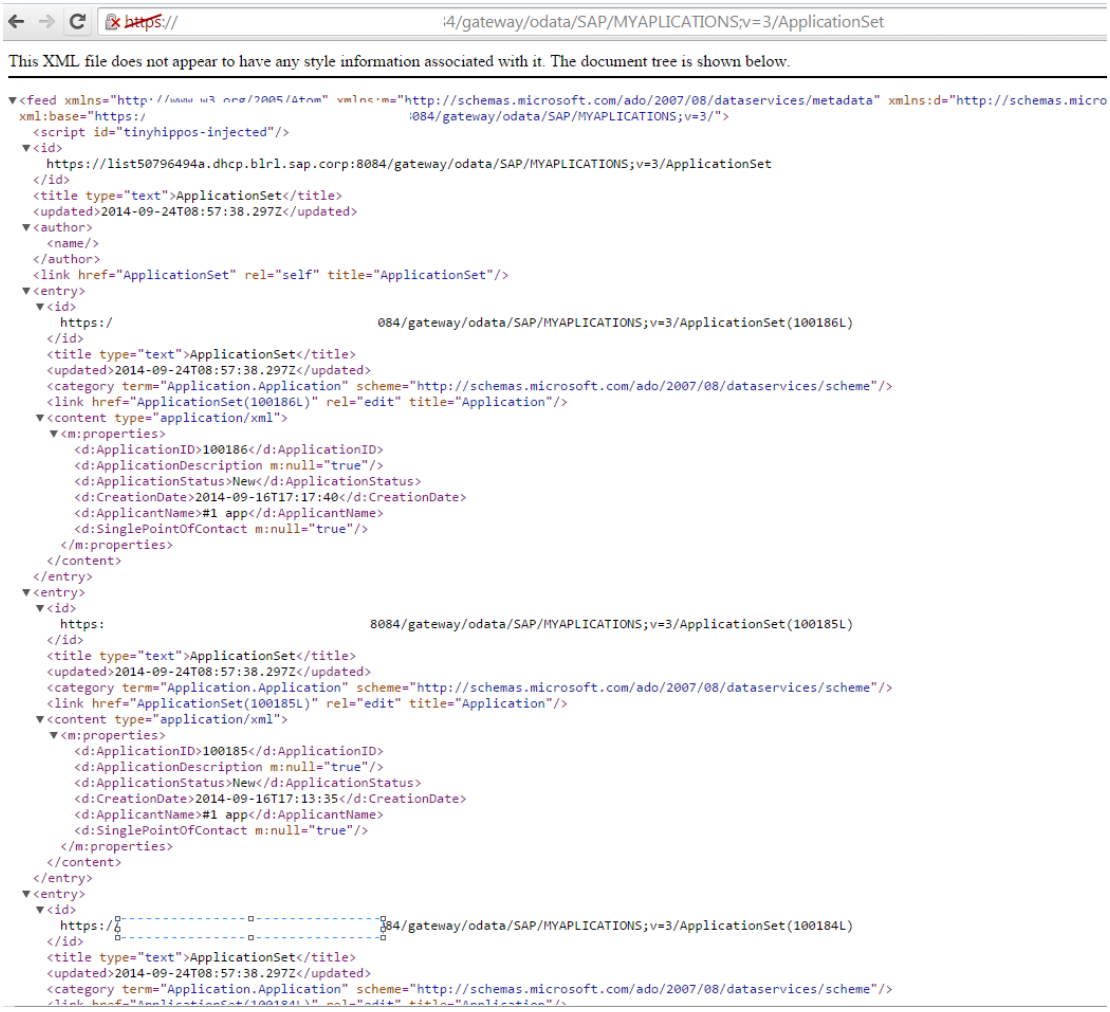
Service Name	Namespace	Description	Version	Activation Status	Default Destination	Service Type	Service Document
APPLICATIONDETAILS	SAP	ApplicationDetails	1	Active		CUSTOM	Open Service Document
APPLICATIONDETAILSWORK	SAP	ApplicationDetailsWork	2	Active		CUSTOM	Open Service Document
ESPMSERVICE	SAP	ESPM Ref Service for demo	1	Active		CUSTOM	Open Service Document
FILTERTEST	SAP	FilterTest	2	Active		CUSTOM	Open Service Document
MYAPLICATIONS	SAP	MyAplications	3	Active		CUSTOM	Open Service Document
MYQUESTIONS	SAP	MyQuestions	1	Active		CUSTOM	Open Service Document

Open the service document by clicking on the “Open Service Document” link

Metadata for the service can be accessed by appending $metadata to the service document

https://<<Host>>:<<port>>/gateway/odata/SAP/<<Model>>;v=3/$metadata

We can execute the service by appending the collection href name to the service document



https://4/gateway/odata/SAP/MYAPLICATIONS;v=3/ApplicationSet

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<feed xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.micro
  xml:base="https:/                                    :8084/gateway/odata/SAP/MYAPLICATIONS;v=3/">
    <script id="tinyhippos-injected"/>
  ▼<id>
      https://list50796494a.dhcp.blrl.sap.corp:8084/gateway/odata/SAP/MYAPLICATIONS;v=3/ApplicationSet
    </id>
    <title type="text">ApplicationSet</title>
    <updated>2014-09-24T08:57:38.297Z</updated>
  ▼<author>
      <name/>
    </author>
    <link href="ApplicationSet" rel="self" title="ApplicationSet"/>
  ▼<entry>
    ▼<id>
        https:/                                    084/gateway/odata/SAP/MYAPLICATIONS;v=3/ApplicationSet(100186L)
      </id>
      <title type="text">ApplicationSet</title>
      <updated>2014-09-24T08:57:38.297Z</updated>
      <category term="Application.Application" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
      <link href="ApplicationSet(100186L)" rel="edit" title="Application"/>
    ▼<content type="application/xml">
      ▼<m:properties>
          <d:ApplicationID>100186</d:ApplicationID>
          <d:ApplicationDescription m:null="true"/>
          <d:ApplicationStatus>New</d:ApplicationStatus>
          <d:CreationDate>2014-09-16T17:17:40</d:CreationDate>
          <d:ApplicantName>#1 app</d:ApplicantName>
          <d:SinglePointOfContact m:null="true"/>
        </m:properties>
      </content>
    </entry>
  ▼<entry>
    ▼<id>
        https:                                    8084/gateway/odata/SAP/MYAPLICATIONS;v=3/ApplicationSet(100185L)
      </id>
      <title type="text">ApplicationSet</title>
      <updated>2014-09-24T08:57:38.297Z</updated>
      <category term="Application.Application" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
      <link href="ApplicationSet(100185L)" rel="edit" title="Application"/>
    ▼<content type="application/xml">
      ▼<m:properties>
          <d:ApplicationID>100185</d:ApplicationID>
          <d:ApplicationDescription m:null="true"/>
          <d:ApplicationStatus>New</d:ApplicationStatus>
          <d:CreationDate>2014-09-16T17:13:35</d:CreationDate>
          <d:ApplicantName>#1 app</d:ApplicantName>
          <d:SinglePointOfContact m:null="true"/>
        </m:properties>
      </content>
    </entry>
  ▼<entry>
    ▼<id>
        https://4                                    84/gateway/odata/SAP/MYAPLICATIONS;v=3/ApplicationSet(100184L)
      </id>
      <title type="text">ApplicationSet</title>
      <updated>2014-09-24T08:57:38.297Z</updated>
      <category term="Application.Application" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
```

Sample Code

Please find the sample custom code below.

```
/*
  Function processRequestData will be called just after the Request
  flow is triggered.
  Implement processRequestData for additional functionalities as
  $TOP, $SKIP, etc.
  Prepare a HashMap and set it to message body. This HashMap has keys as
  parameter names for the required web service operation, and values as
  the values for those parameters.
 */
function processRequestData(message) {


        importPackage(com.sap.gateway.ip.core.customdev.logging);
        importPackage(com.sap.gateway.ip.core.customdev.util);
        importPackage(org.apache.olingo.odata2.api.uri);
        importPackage(java.util);
        importPackage(com.sap.gateway.core.ip.component.commons);
        importPackage(com.sap.gateway.ip.core.customdev.api);

        var response = new LinkedHashMap();
        message.setHeader("Authorization", "Basic c3BjMDE6YWJjZDEyMzQ=");

        map = new LinkedHashMap();
        map.put("applicationID", "100154");
```

```
        partneHashMap = new LinkedHashMap();
        partneHashMap.put("arg0", map);
        message.setBody(partneHashMap);

        return message;
}




/*
  Gets the SOAP request XML here. This method can be used if extra headers
  need to be added to the XML request.
 */
function processRequestXML(message) {

        return message;
}

/*
  Gets the web service response XML. If the web service returns values which are not mapped to Edm types,
  those values can be filtered out here eg :- Inline count, eTag, etc.
 */
function processResponseXML(message) {



        return message;
}

/*
  Implement processResponseResult to modify the ResultSet returned from the
  Database to handle scenarios such as Deltatoken and Tombstone.
  In this method, the integration developer gets the response after it is converted to a HashMap, which
  can be accessed through the message body.

  DELTATOKEN: Choose/generate a deltatoken and sets this value as a header in
  the message object. TOMBSTONE: The ResultSet HashMap needs to be split into
```

```
  two HashMaps. One for result entities and the other for deleted items. A user
  defined algorithm can be used for splitting. Result entites HashMap needs to
  be set as message body and deleted entities HashMap has to be set as the
  header(shown in Sample Implementation) in the message object.
 */
function processResponseData(message) {


        return message;
}
```

Copyright