Generating Web Service Manually for a Rules Project (using XSD)

Applies to

SAP NetWeaver Composition Environment 7.1.1
For more information, visit the Business Rules Management homepage.

Summary

This article helps you in creating a web service which can be used in any application to invoke the required set of Business Rules.

In this article, an example is showcased for generating a web service manually, for a Rules Project which has been created by using an XSD.

Author: Arti Gopalan

Company: SAP Labs India Pvt. Ltd.

Created on: 28 August 2008

Table of Contents

Introduction

Business rules describe the operations and constraints that are applicable to an organization and thus help in guiding the business processes. NetWeaver allows you to cumulate these sets of business rules by making use of the rules composer DC.

Once these rules have been collected in a rules DC you can use them in any of your applications as required. One way of doing the same, is to expose these Business Rules (or Business Rulesets) as a web service.

This article helps you in creating a web service which can be used in any application to invoke the required set of Business Rules. Here, an example is showcased for generating a web service manually, for a Rules Project which has been created by using an XSD.

Note: The example has been provided at this location:
https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/70a3f50b-9651-2b10-5dbb-bbf308145252

Generating Web Service DCs

You can generate a web service manually for a rules project (using XSD as its Business Object).

1. Extract the DCs - *generic_xsd_ws* (a WebModule DC) and *generic_xsd_ear* (an Enterprise Application DC) from the zip file.
2. Copy these DCs to the path of the NetWeaver Developer Studio workspace location as shown below:
   Example-
   <Workspace>.jdi\LocalDevelopment\DCs\demo.sap.com
3. In SAP NetWeaver Developer Studio, choose Window -> Open Perspective -> Development Infrastructure.
4. In the *Component Browser* view, expand *Local Development* node and *MyComponents [demo.sap.com]*

You should see the two DCs copied into the Workspace's DC under *MyComponents [demo.sap.com]* node.

Note: If you are not able to view the *warranty_claims_demo* node, right click and refresh the *Componenet Browser* view.

1. In the context menu of the DCs, choose *Sync / Create Project -> Create Project*.
2. In the *Create DC Projects* screen that appears, choose *OK*.

In the *Confirm Perspective Screen* that appears, choose *Yes.*

Customizing the DCs

List of modifications to be made -

1. Open the Web Module DC - generic_xsd_ws .
2. A class - GenericWebService.java – has already been created in this DC.
3. The web service provider information has been defined in the class. If required, changes can be made to the following parameters – serviceName, portName, targetNamespace and wsdlLocation.

For example: In the DCs provided here, these parameters have been set as shown in the following figure -

```
package com.sap.wsgenerator;

import java.io.ByteArrayOutputStream;

@WebServiceProvider(serviceName = "GenericService", portName = "GenericServicePort",
        targetNamespace = "http://example.com/", wsdlLocation = "WEB-INF/wsdl/RuleService.wsdl")
@ServiceMode(value = Mode.MESSAGE)
public class GenericWebService implements Provider<SOAPMessage> {
```

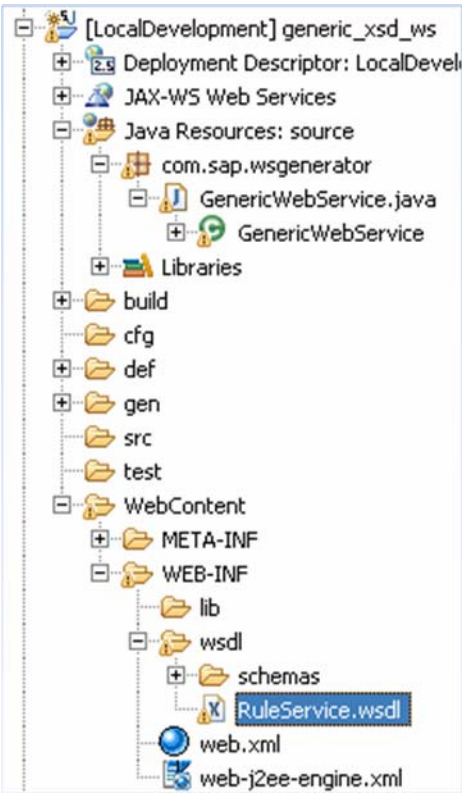
Note: A wsdl –RuleService.wsdl - has been provided in this DC

1. Open this class and make changes to the *projectname* and the *rulesetname* defined in the project.

```
public GenericWebService() {
    try {
        projectName = "<Project Name>";
        rulesetName = "<Ruleset Name>";
        documentBuilder = null;
    } catch (Exception e) {
```

2. A wsdl has been created in this DC – *RuleService.wsdl*
3. The wsdl contains the information about the xsd, which is being used in the Rules Project (for which the web service is being created)
4. The XML Schemas used in the Rules Project has been placed inside the folder *WEB-INF* > *wsdl* > *schemas*

5. The DC when viewed in the Project Explorer looks like this –



Customizing the wsdl file

The following changes have to be made to the following –

- The *xsd Namespace* mentioned in the wsdl.
- The *Root Element* mentioned.
- The *Service Name* and the *Service Port Name* (if any changes are made to these parameters in the java class)

For example:

In this demo provided here:

1. the xsd Namespace has been set to – *http://www.example.org/CallCharge*
2. the Root Element in the xsd is – *CallCharges*
3. the Service Name and the Service Port Name have been set to – *GenericService* and *GenericServicePort*

Building and Deploying the DCs

Prerequisites

You should have a running instance of SAP AS, and should have configured the SAP NetWeaver Developer Studio with this instance.

Building the Web Module DC

1. In the Project Explorer view, from the context menu of the generic_xsd_ws node, choose Development Component > Build
2. In the dialog box that appears, make sure the generic_xsd_ws checkbox is selected and choose OK

To check if the build has happened successfully, check the Infrastructure Console.

Note: If the Infrastructure Console is not open, choose Window > Show View > Other and in the dialog box that appears, expand the Development Infrastructure node and choose Infrastructure Console and then choose OK.

Building and Deploying the Enterprise Application DC

1. In the Project Explorer view, in the context menu of the generic_xsd_ear node, choose Development Component > Build
2. In the dialog box that appears, make sure the generic_xsd_ear checkbox is selected and choose OK

To check if the build has happened successfully, check the *Infrastructure Console*.

Note: If the *Infrastructure Console* is not open, choose *Window* > *Show View* > *Other* and in the dialog box that appears, expand the *Development Infrastructure* node and choose *Infrastructure Console* and then choose *OK*.

1. In the context menu of generic_xsd_ear node, choose Development Component > Deploy
2. In the Deploy DCs dialog box that appears, select the generic_xsd_ear checkbox and choose OK

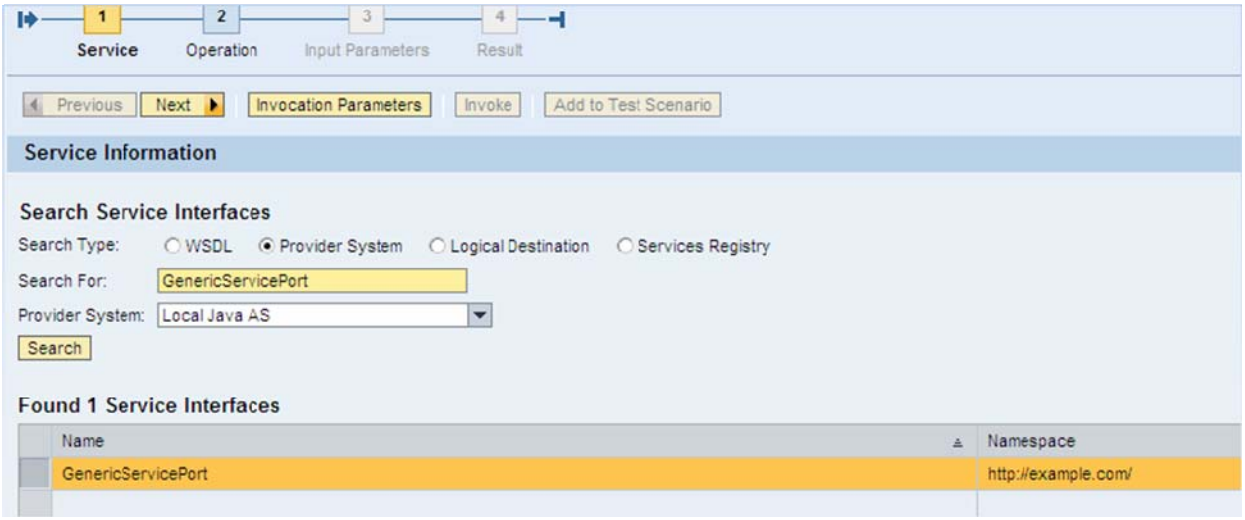
Note: Open the Infrastructure Console, to check if the deploy has happened successfully.

Getting the WSDL from the WS Navigator

Make sure that the SAP NetWeaver Application Server is running.

You can test if the web service has been deployed correctly into the application server by doing the following-

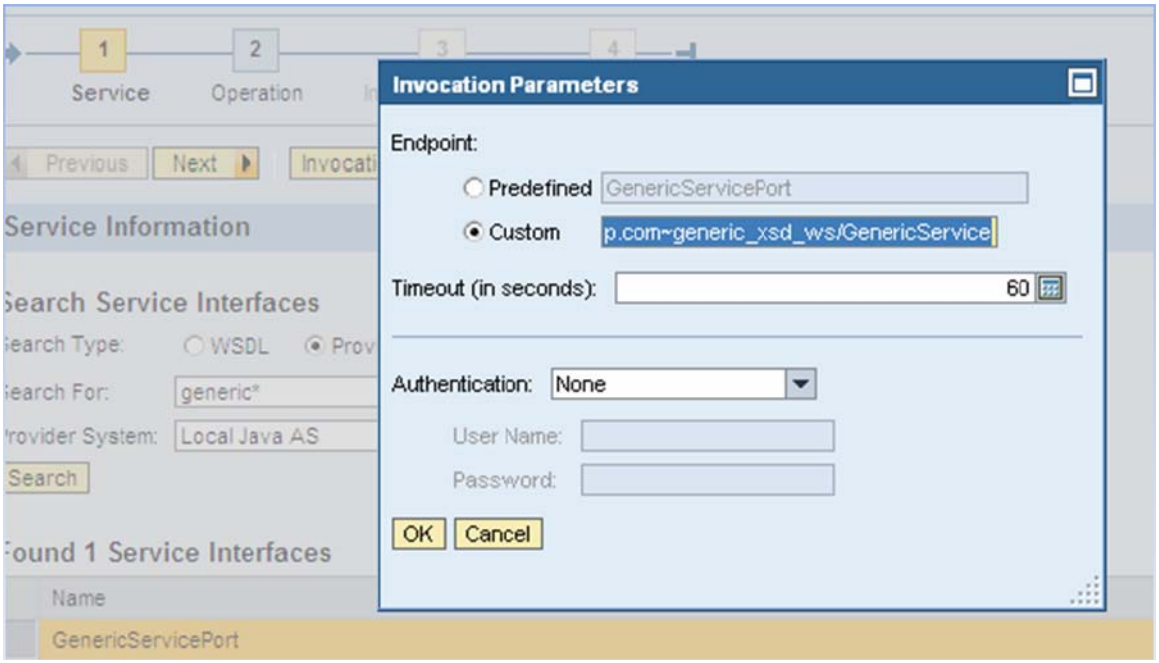
1. Open the browser and enter the Application Server Address followed by the port number.
   You should see the *SAP NetWeaver Application Server Java* page.
2. Choose the *Web Services Navigator* icon.
3. Enter the user id and password in the *User ID* and *Password* fields. Choose *Log On*.
4. In the *SAP NetWeaver WS Navigator* page that appears, select the *Provider System* radio button.
5. Enter the service name in the *Search For* text box (as set in the java class – here in the example, we have set the port name to be *GenericServicePort*) and choose *Search* button. Under the *Service Interfaces* section, you can view the service as shown below.



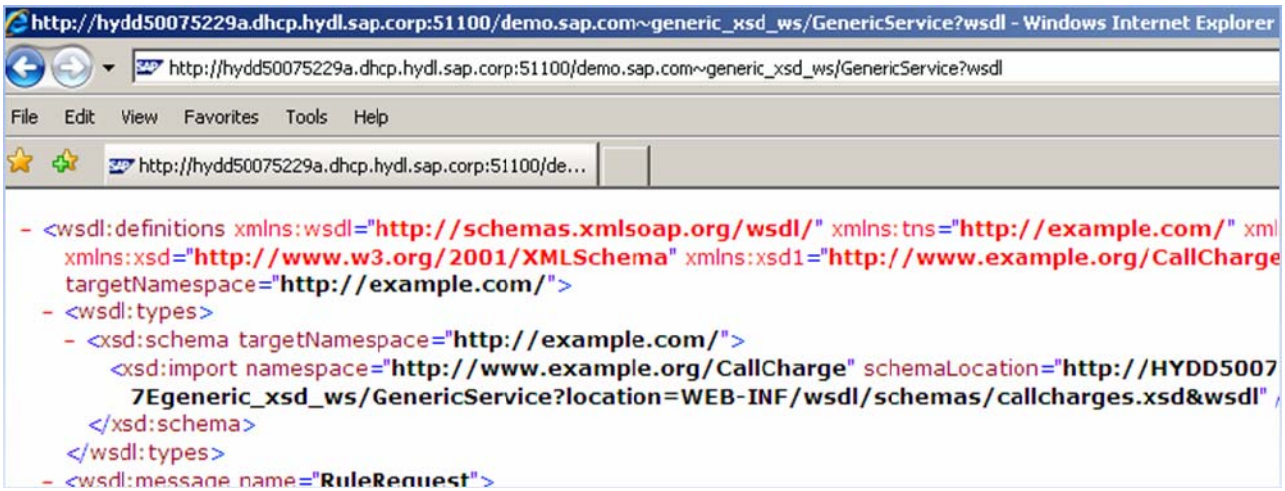
6. To get the WSDL location, choose the *GenericServicePort* and choose *Next* button.
7. In the page that appears, choose the *InvokeRule* method and choose *Next*.
8. Set the Paramenters and choose *Next*.
   You can view the Results.

To view the wsdl –

1. In the wsnavigator portal, search for the web service port name, as set in the java class. (Here in the example, we have set the port name to be GenericServicePort)
2. Choose the GenericServicePort and choose Invocation Parameters button.
3. In the Invocation Parameters screen that appears, select the Custom radio button and copy the link.



4. Choose Cancel.
5. Open the browser and use the link followed by ?wsdl to access the WSDL file as shown below.



Copyright