

Applies to:

SAP NetWeaver Business Rules Management

Summary

This tutorial helps you get started with the rules composer. A business use case has been created for the purpose.

Given details such as connection provider, type of line/connection and connection destination, the long distance phone rate must be calculated.

This tutorial guides you to create rules to calculate the long distance phone rate given a set of criteria.

Author(s): Arti Gopalan and Rupa Rajagopalan

Company: SAP Labs India

Created on: 28th May 2008

Table of Contents

Prerequisites.....	3
Knowledge Required	3
Software Requirements	3
Procedure	3
Creating the Rules Composer DC.....	3
Creating the XML Schema.....	4
Adding XSD Elements.....	5
Renaming the XSD Aliases (Optional)	7
Creating the Ruleset	7
Creating the Decision Table.....	7
Adding Condition and Action Values.....	10
Creating the Rule.....	11
Deploying the Rules	13
Executing the Rules.....	14
Creating the Web Module.....	14
Adding Dependency to the Web Module	14
Creating EngineInvoker.java.....	15
Creating the XMLHelper.java.....	18
Creating the CallCharges.jsp.....	20
Creating the index.jsp	22
Creating the invoker.jsp.....	23
Creating the Enterprise Application.....	25
Adding Dependency to the Enterprise Application	25
Creating application.xml	25
Building and Deploying	26
Running the Web Module	27

Prerequisites

Knowledge Required

- You have basic knowledge in rules modeling
- You are familiar with Business Rules Management System

Software Requirements

- You work in the SAP NetWeaver Developer Studio
- Your SAP NetWeaver Developer Studio version includes the Rules Composer perspective
- You should have a running instance of SAP AS, and should have configured the SAP NetWeaver Developer Studio with this instance

Note

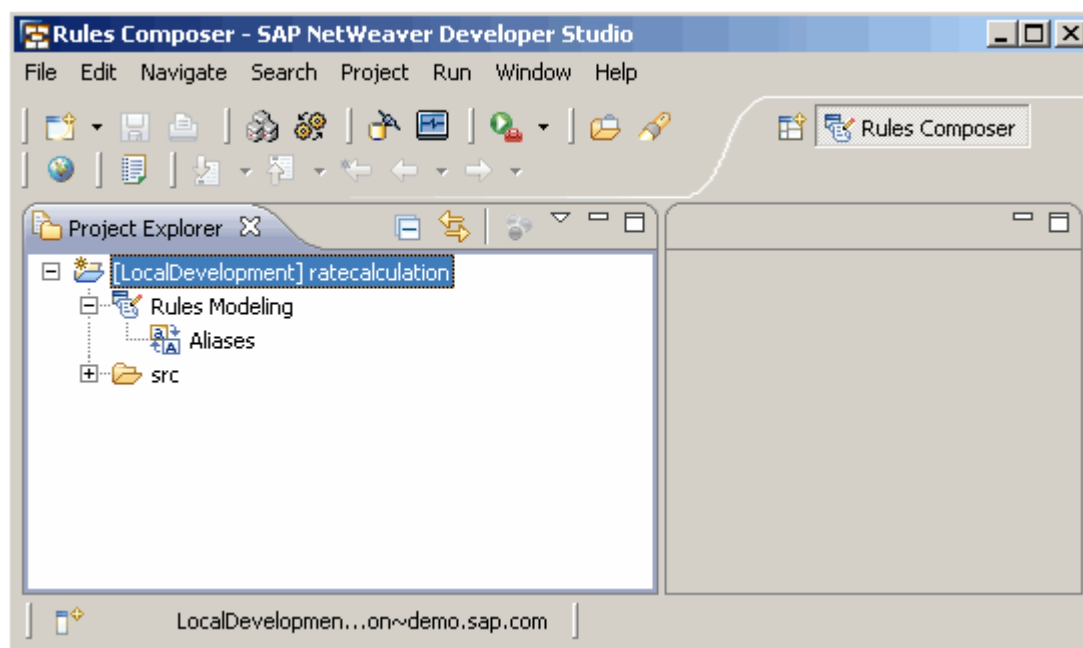
In the SAP NetWeaver Developer Studio, choose *Window -> Open Perspective -> Other*. In the dialog box that appears, choose *Rules Composer* and choose OK.

Procedure

Creating the Rules Composer DC

1. In the SAP NetWeaver Developer Studio, choose *File -> New -> Project*.
2. In the wizard that appears, expand the *Rules Composer* node and choose *Rules Composer Development Component*. Choose *Next*.
3. In the screen that appears, choose the software component where you want to create the DC.
For example the software component could be *MyComponents [demo.sap.com]* under the *Local Development* node. Choose *Next*.
4. In the screen that appears, enter **ratecalculation** in the *Name* field and choose *Finish*.

You should see the Rules Composer DC: *ratecalculation* node in the *Project Explorer* view as shown below:



Creating the XML Schema

Procedure

1. In the *Project Explorer* view, expand the *src* node and in the context menu of the *wsdl* node, choose *New > Other*.
2. In the wizard that appears, expand the *XML* node and choose *XML Schema*. Choose *Next*.
3. In the screen that appears, enter *callcharges.xsd* in the *File Name* field. Choose *Finish*.

In the *Project Explorer* view, you should see the *callcharges.xsd* under the *wsdl* node.

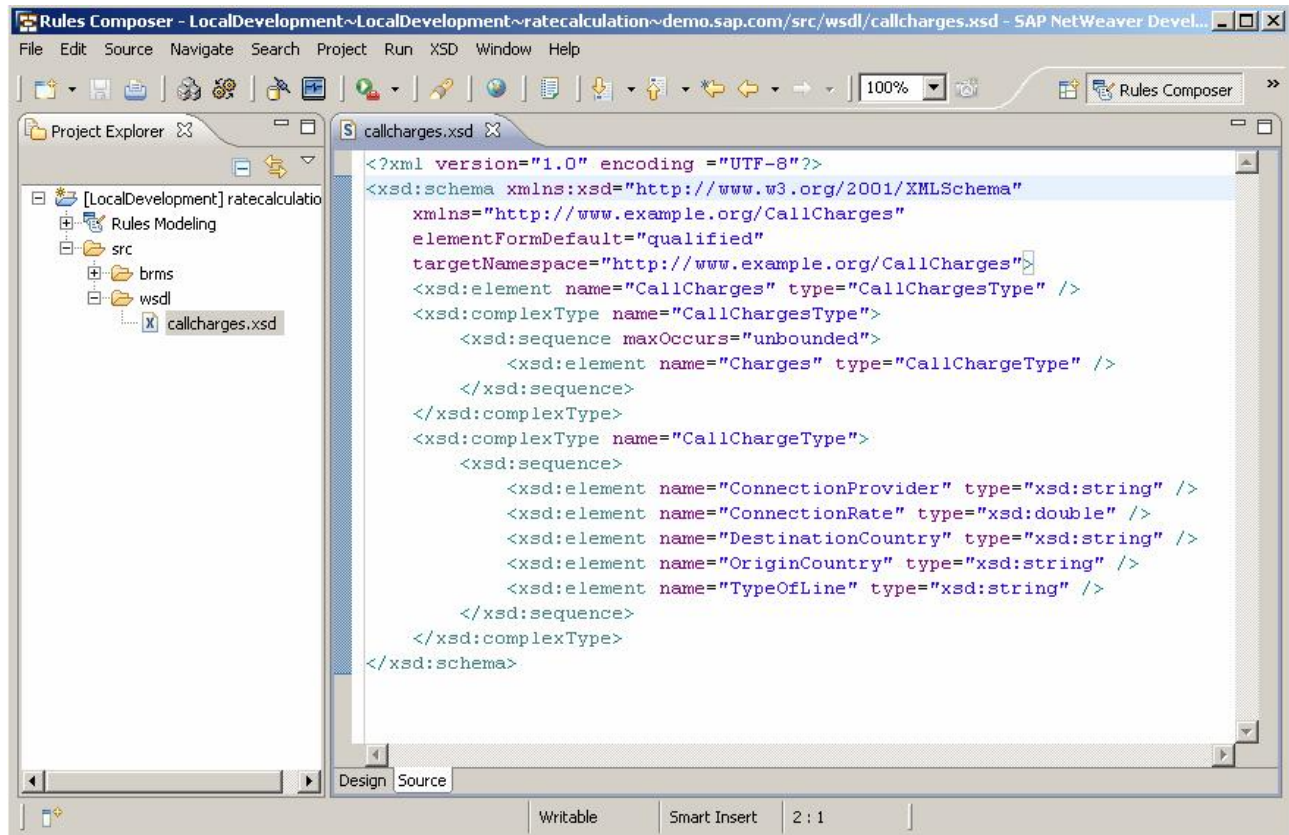
4. In the *callcharges.xsd* window that appears choose the *Source* tab at the bottom.
5. In the tab page that appears delete all existing content and copy the following in the *Source* tab page:

Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.example.org/CallCharges"
elementFormDefault="qualified"
targetNamespace="http://www.example.org/CallCharges">
<xsd:element name="CallCharges" type="CallChargesType"/>
  <xsd:complexType name="CallChargesType">
<xsd:sequence maxOccurs="unbounded">
<xsd:element name="Charges" type="CallChargeType"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CallChargeType">
<xsd:sequence>
<xsd:element name="ConnectionProvider" type="xsd:string"/>
<xsd:element name="ConnectionRate" type="xsd:double"/>
<xsd:element name="DestinationCountry" type="xsd:string"/>
<xsd:element name="OriginCountry" type="xsd:string"/>
<xsd:element name="TypeOfLine" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

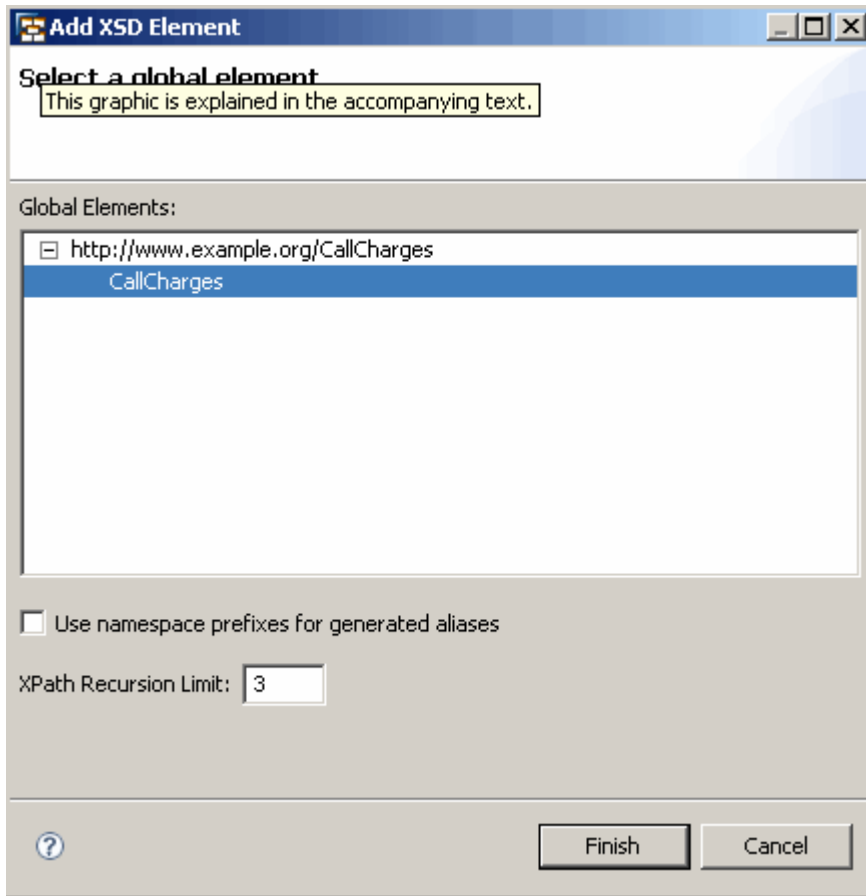
6. Press **Ctrl+Shift+F**.
7. Save the changes

The result must be as shown below:

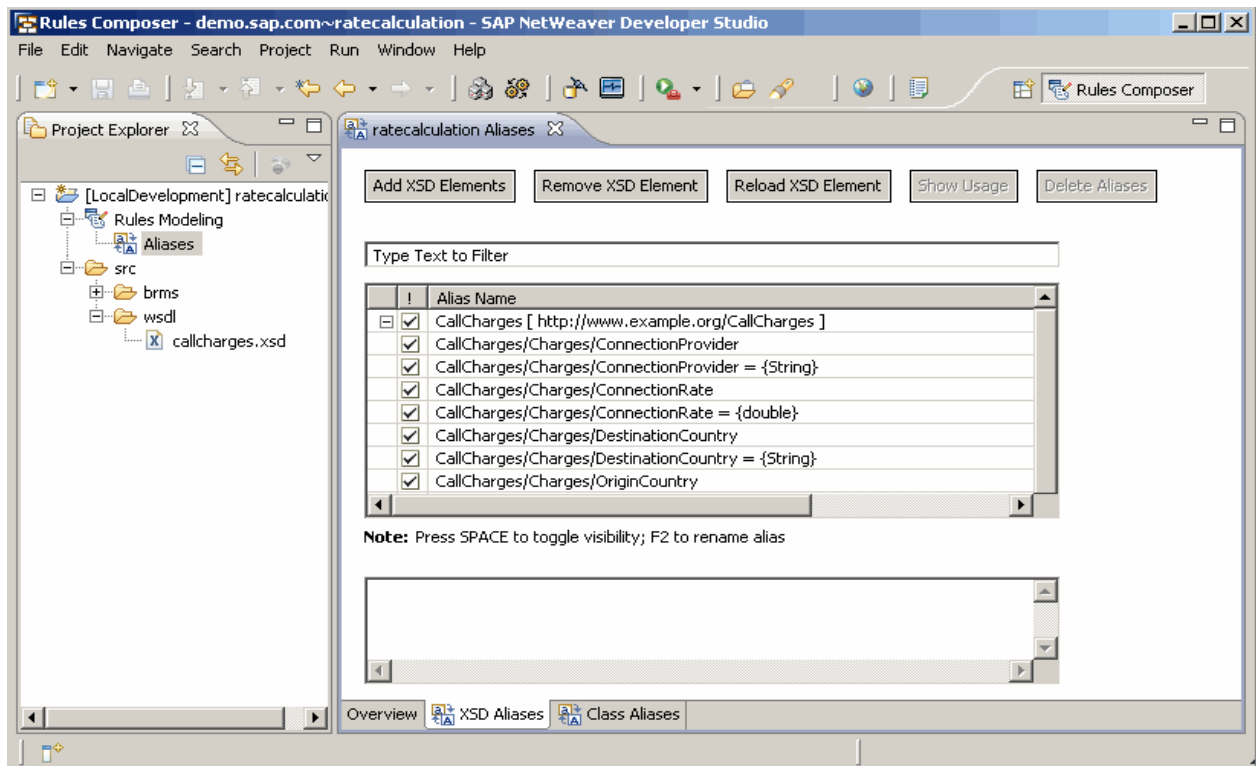


Adding XSD Elements

1. In the *Project Explorer* view, expand the *Rules Modeling* node and double-click the *Aliases* node.
2. In the Project Aliases Editor that appears, choose the *XSD Aliases* tab and in the tab page that appears, choose the *Add XSD Elements* tab.
3. In the dialog box that appears, expand the *http://www.example.org/CallCharges* node and select *CallCharges* as shown below:



4. Choose *Finish*.
5. In the *Alias Name* table select all the XML schema element checkboxes as shown below:



6. Save the changes.

Renaming the XSD Aliases (Optional)

1. In the *Alias Name* table, click each of the aliases. The aliases become editable. Enter an alternative name for the alias.

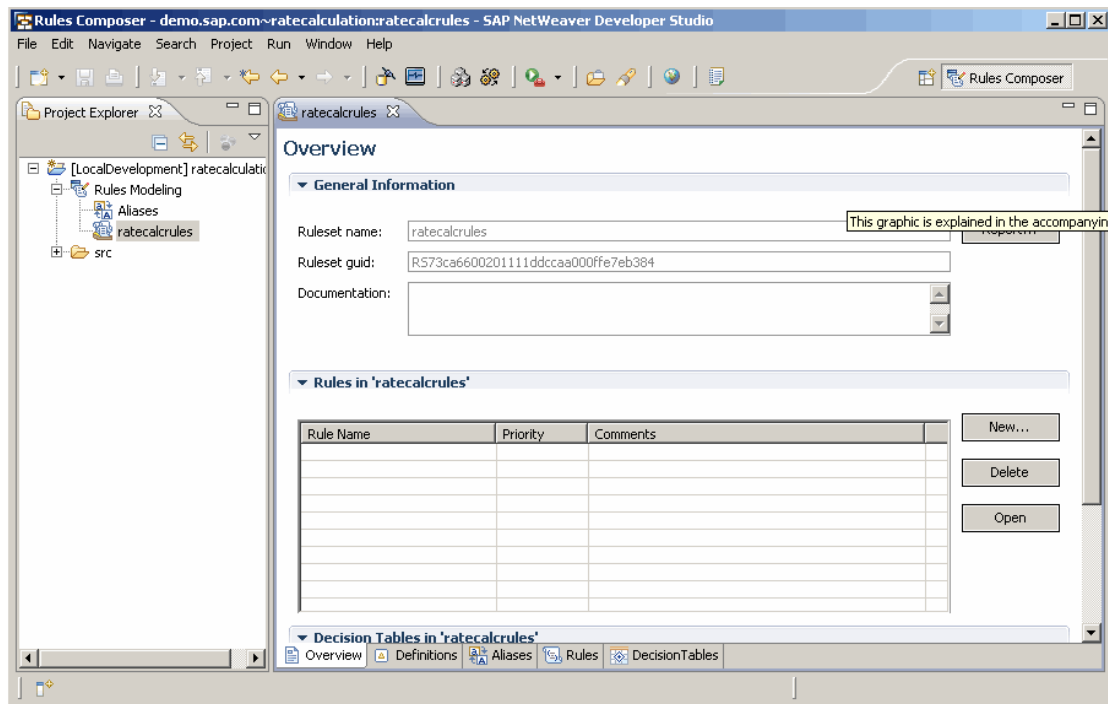
Alias Name	Rename as
CallCharges/Charges/ConnectionProvider	Connection Provider
CallCharges/Charges/ConnectionRate = {double}	Set Rate = {double}
CallCharges/Charges/DestinationCountry	Destination Country
CallCharges/Charges/TypeOfLine	Type Of Line

Creating the Ruleset

Procedure

1. In the *Project Explorer* view, expand the *ratecalculation* node and in the context menu of the *Rules Modeling* node, choose *New Ruleset*.
2. In the dialog box that appears, enter *ratecalcrules* in the field. Choose *OK*.

In the *Project Explorer* view you should see the *ratecalcrules* node under the *Rules Modeling* node.



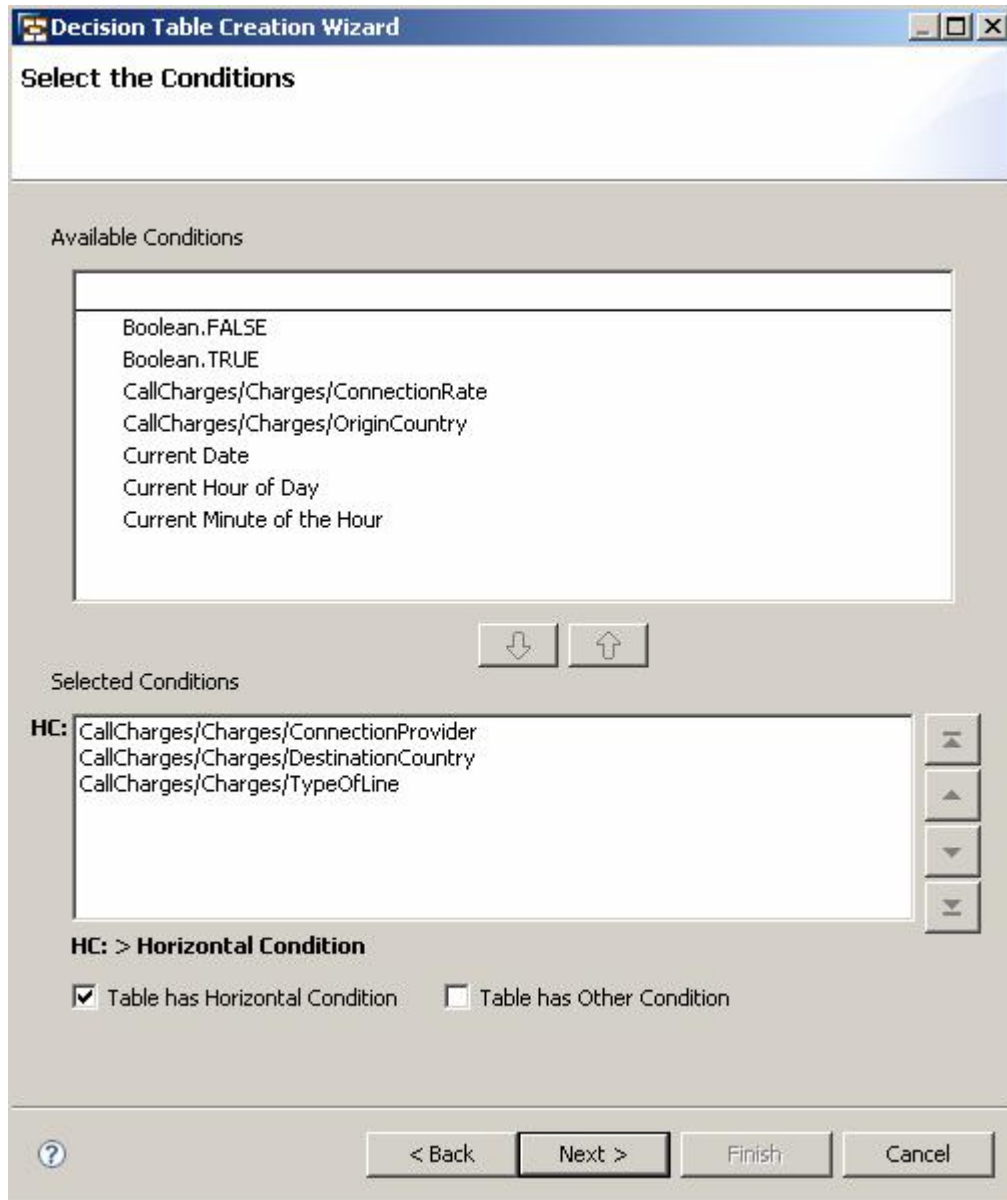
Creating the Decision Table

You can capture the criteria and the corresponding rates for long distance phone calls in a single table.

Procedure

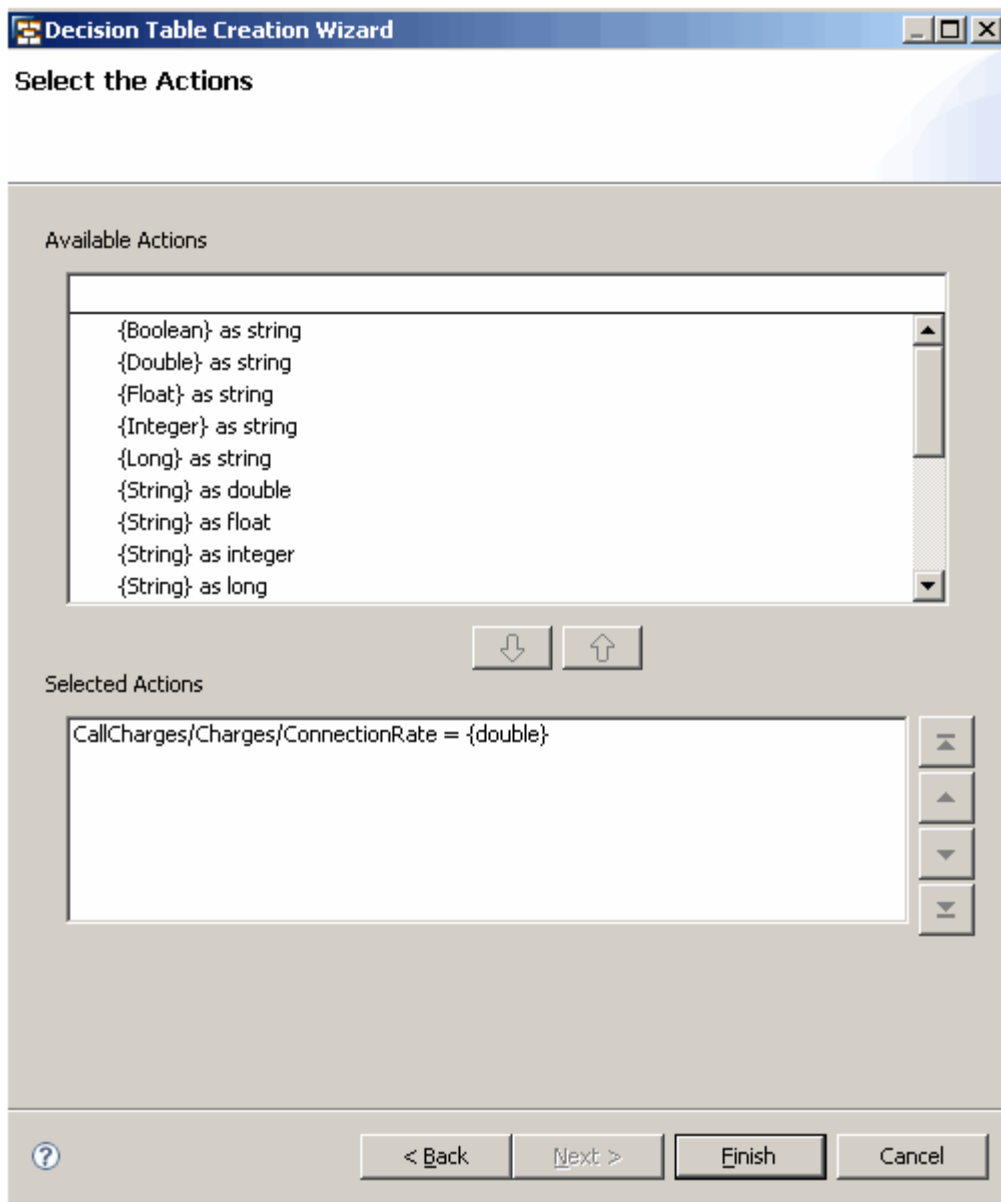
1. In the *Project Explorer* view, expand the *ratecalculation* node, the *Rules Modeling* node and in the context menu of the *ratecalcrules* node, choose *New Decision Table*.
2. In the *Decision Table Creation Wizard* that appears, enter *ratestable* in the *Decision Table Name* field and choose *Next*.
3. On the *Select the Conditions* screen, double-click the following aliases in the *Available Conditions* section:
 - *CallCharges/Charges/ConnectionProvider*
 - *CallCharges/Charges/DestinationCountry*
 - *CallCharges/Charges/TypeOfLine*

The aliases appear in the *Selected Conditions* section as follows:



4. Select the *Table has Horizontal Condition* checkbox. Choose *Next*.
5. On the *Select the Actions* screen, double-click *CallCharges/Charges/ConnectionRate = {double}* in the *Available Actions* section.

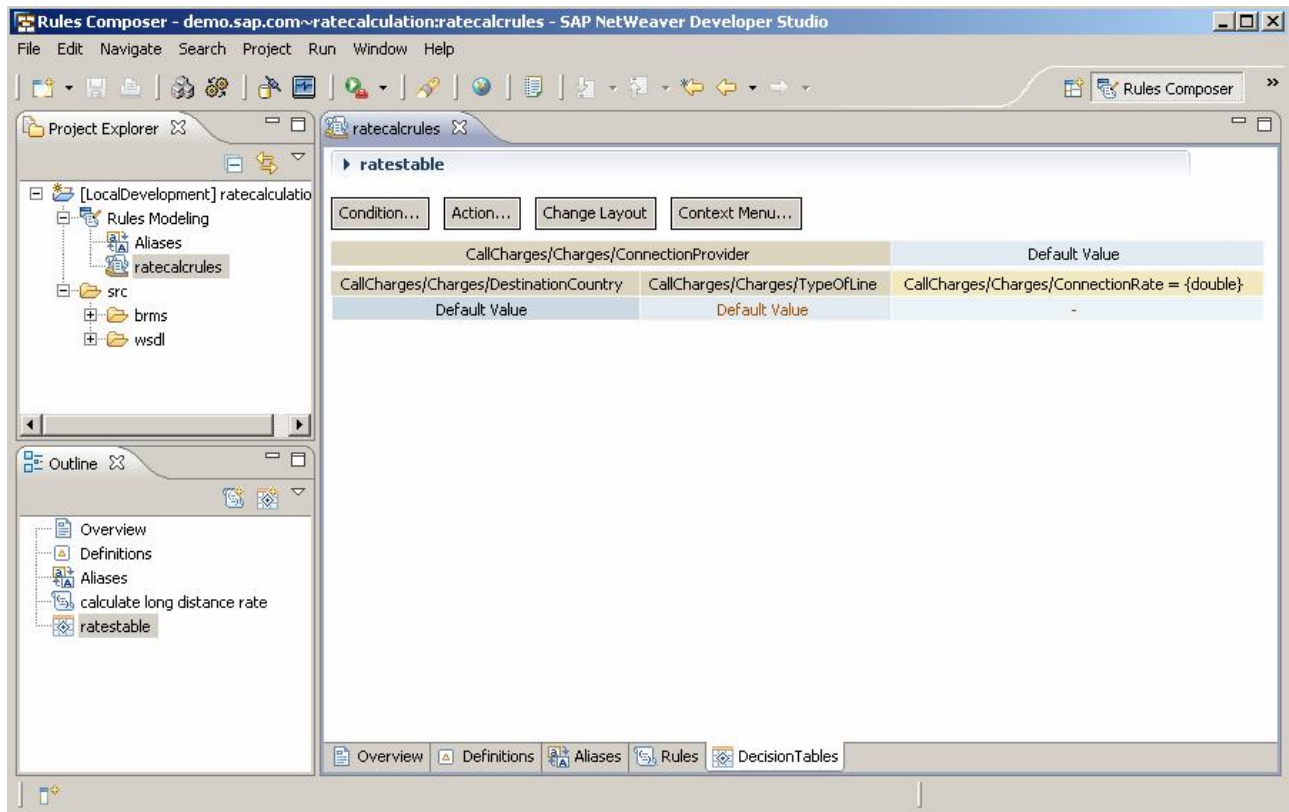
The alias appears in the *Selected Actions* section as follows:



6. Choose *Finish*.
7. Save the changes.

In the *Outline* view, the *ratestable* node appears.

The Decision Tables Editor appears as shown below:



Adding Condition and Action Values

1. In the Decision Table Editor that appears, double-click each of the cells under the condition headers (*CallCharges/Charges/Connection Provider*, *CallCharges/Charges/Destination Country*, and *CallCharges/Charges/Type Of Line*) and enter the values.

Note: You can also choose *Add Condition Value* in the context menu of the cells under the condition headers and in the dialog box that appears enter all the values in the available rows.

2. Double-click each cell under the action header (*CallCharges/Charges/ConnectionRate = {double}*) in the Decision Table, enter a value and press **Enter** key.

Refer to the Decision Table below for the condition and action values you need to enter:

CallCharges/Charges/ConnectionProvider		WorldProvider	OnlyEuroProvider	OnlySwitchProvider
EuroCountry1	Dedicated	0.47	0.47	0
	Switched	0.49	0.485	0.53
AsianCountry	Dedicated	0.87	0	0
	Switched	0.91	0	0.88
EuroCountry2	Dedicated	0.195	0.15	0
	Switched	0.20	0.16	0.216

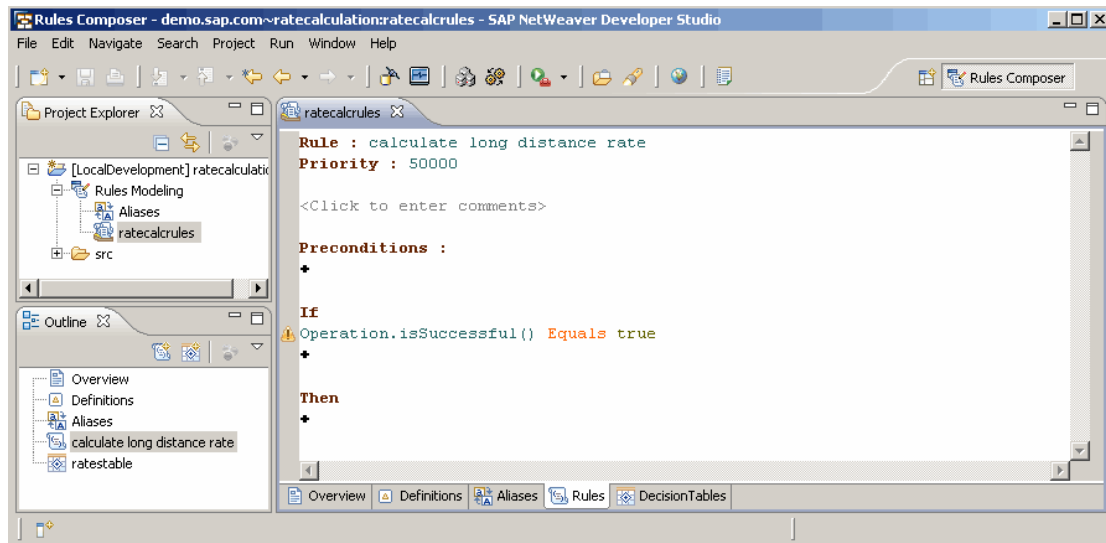
Creating the Rule

You need to create a rule called `calculate long distance rate` that evaluates the `ratestable`.

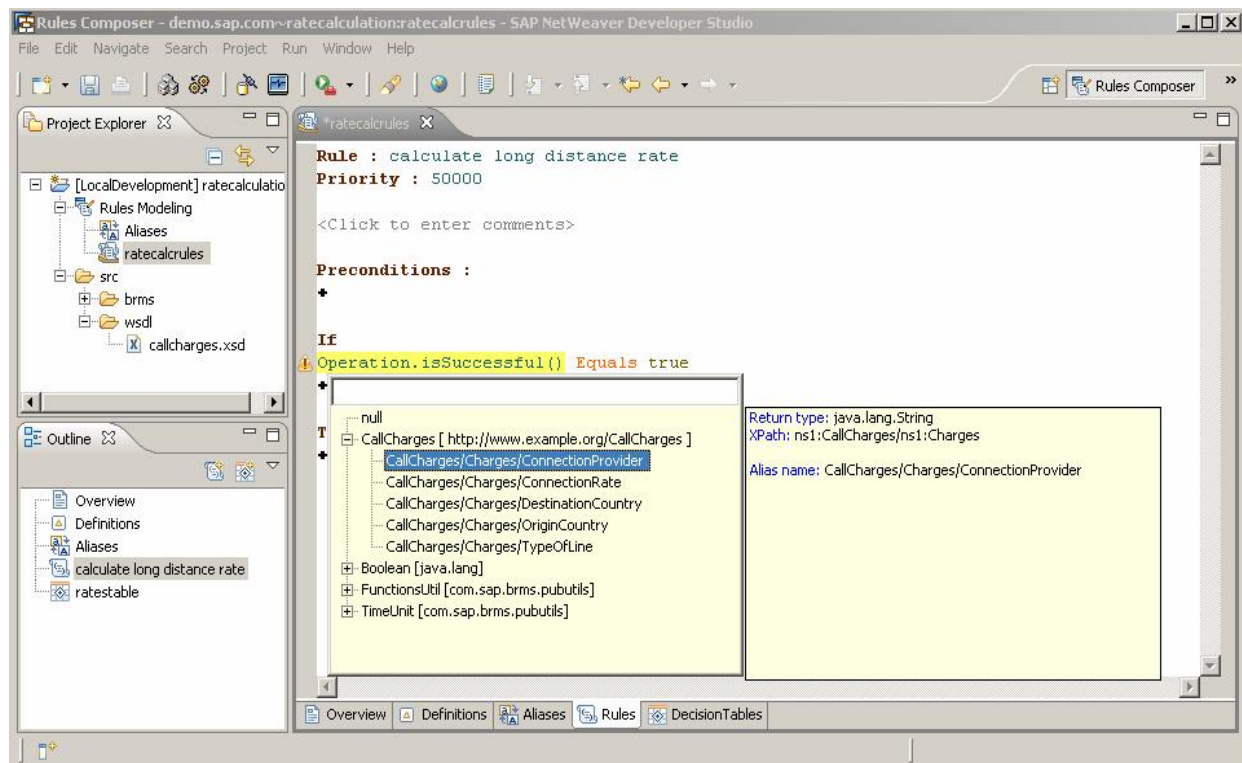
Procedure

1. In the *Project Explorer* view, expand the `ratecalculation` node, the *Rules Modeling* node and in the context menu of the `ratecalcrules` node, choose *New Rule*.
2. In the dialog box that appears, enter `calculate long distance rate` in the field and choose *OK*.
3. In the Rule Editor that appears, under *If* section, choose the *Add* icon.

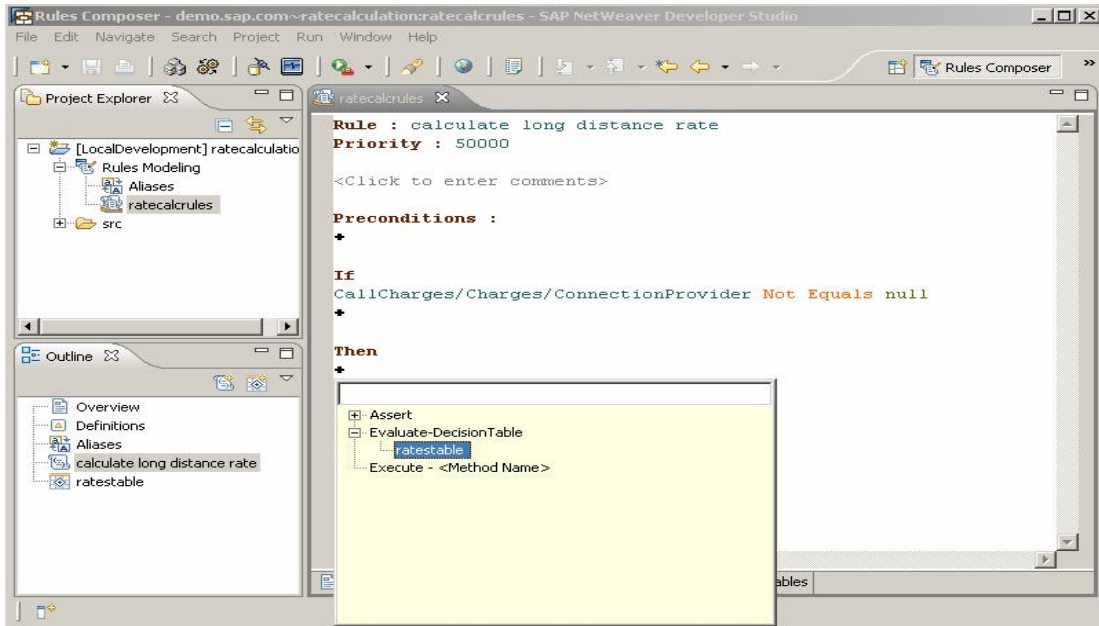
The default condition: `Operation.isSuccessful Equals true` appears as shown below:



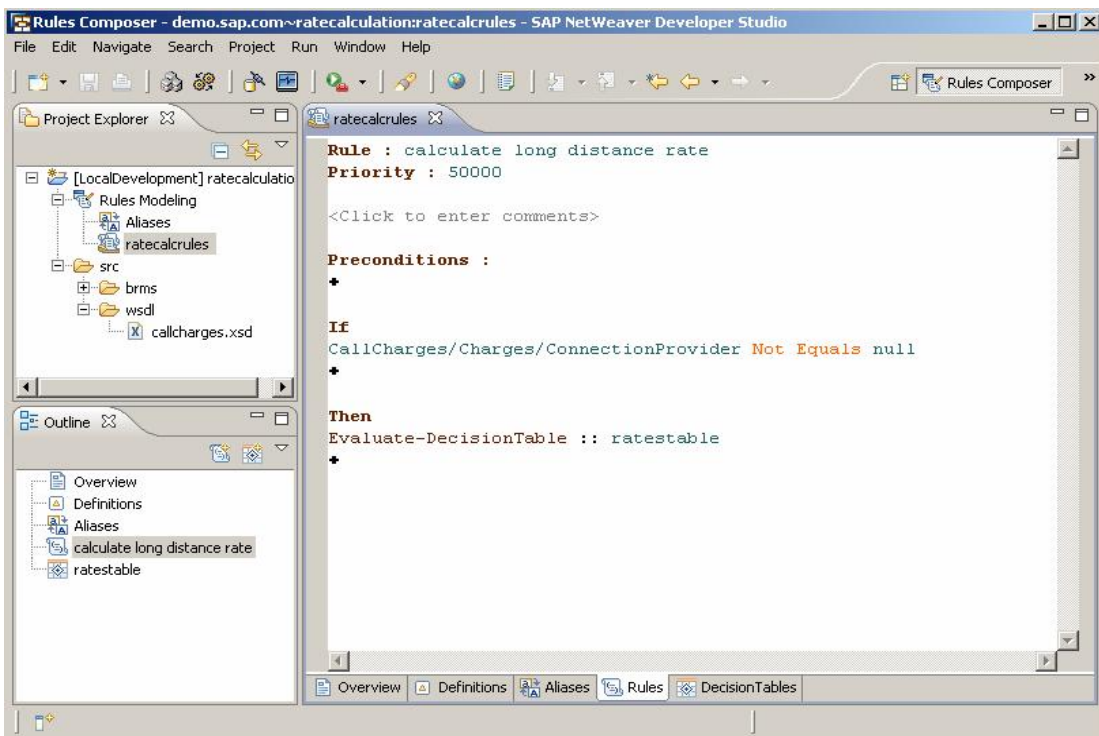
4. Edit the default condition as follows:
 1. Choose the LHS value: *Operation.isSuccessful* and in the drop down menu choose *CallCharges/Charges/Connection Provider* as shown below:



2. Choose the comparator: *Equals* and in the drop down menu choose *Not Equals*.
 3. Choose the RHS value: *Default Value* and in the inline textbox enter *null*.
5. Under *Then* section, choose the icon that looks like a plus sign and in the drop down menu that appears, expand the *Evaluate Decision Table* node and choose *ratestable* as shown below:



The result must be as shown below:



6. Save the changes.

Deploying the Rules

Prerequisites

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

Procedure

1. In the *Project Explorer* view, in the context menu of the *ratecalculation* node, choose *Development Component > Build*.
2. In the dialog box that appears, make sure the *ratecalculation* 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*.

3. In the context menu of the *ratecalculation* node, choose *Development Component > Deploy*.
4. In the *Deploy DCs* dialog box that appears, select the *ratecalculation* checkbox and choose *OK*.

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

A dialog box saying *Deploy finished successfully* appears.

Executing the Rules

Procedure

Creating the Web Module

1. In the SAP NetWeaver Developer Studio, choose *File > New > Project*.
2. In the wizard that appears, expand the *Development Infrastructure* node and choose *Development Component*. Choose *Next*.
3. In the screen that appears, expand the *J2EE* node and choose *Web Module*. Choose *Next*.
4. In the screen that appears, choose the software component where you want to create the DCs. For example expand the *'Local Development'* node and choose *MyComponents [demo.sap.com]*. Choose *Next*.
5. In the screen that appears, enter `rates` in the *Name* field. Choose *Finish*.

The *Java EE* perspective opens and in the *Project Explorer* view, you should see the *rates* node.

Note

- If the *Project Explorer* view does not open, choose *Window > Show View > Other*.
- In the dialog box that appears, expand the *General* node and choose *Project Explorer*.

Adding Dependency to the Web Module

1. Choose *Window > Open Perspective > Other*.
2. In the dialog box that appears, choose *Development Infrastructure*. Choose *OK*.

Note

- If the *Component Browser* view does not open, choose *Window > Show View > Other*.
- In the dialog box that appears, expand the *Development Infrastructure* node and choose *Component Browser*. Choose *OK*.

3. In the *Component Browser* view, expand the *MyComponents[demo.sap.com]* node and choose the *rates* node.

Note: If the *Component Properties* view does not open, choose *Window -> Show View -> Other*. In the dialog box that appears, expand the *Development Infrastructure* node and choose *Component Properties*. Choose *OK*.

4. In the *Component Properties* view, choose the *Dependencies* tab.
5. Choose the *Add* button and in the wizard that appears, expand the *BRMS-FACADE[sap.com]* node and select the *tc/brms/facade* checkbox. Choose *Next*.
6. In the screen that appears, select the *Design Time*, *Deploy Time*, *Run Time* checkboxes. Choose *Finish*.

Note

- In the context menu of the *rates* node, choose *Sync/Create Project > Sync Used DCs*
- In the dialog box that appears, choose *OK*.

Creating EngineInvoker.java

Make sure that you are in the *Java EE* perspective.

1. Expand the web module: *rates* node and in the context menu of the *Java Resources: source* node, choose *New > Other*
2. In the wizard that appears, expand the *Java* node and choose *Package*. Choose *Next*.
3. In the screen that appears, enter `com.sap.helper` in the *Name* field.
4. Choose *Finish*.
5. In the context menu of *com.sap.helper*, choose *New > Other*
6. In the wizard that appears, choose *Class*. Choose *Next*.
7. In the screen that appears, enter `EngineInvoker` in the *Name* field.

You should see the *EngineInvoker.java* window.

8. Delete all existing lines and copy the following lines into the window:

Syntax

```
package com.sap.helper;

import com.sap.brms.qrules.ejb.RuleEngineHome;
import com.sap.brms.qrules.engine.RuleEngine;
import com.sap.brms.qrules.xml.XMLObject;
import com.sap.brms.qrules.xml.XMLObjectFactory;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.StringReader;
import java.io.StringWriter;
import java.util.*;
import javax.naming.InitialContext;
```

```

import javax.rmi.PortableRemoteObject;
public class EngineInvoker
{
    private static String jndiName = "com.sap.brms.RuleEngine";
    private static String payloadSeparator;
    private static String ret_payloadSeparator;
    private static final String PROPS_FILE = "engine.properties";
    public EngineInvoker()
    {
    }
    public static RuleEngine getRuleEngine()
        throws Exception
    {
        InitialContext context = new InitialContext();
        Object obj = context.lookup(jndiName);
        RuleEngineHome home =
(RuleEngineHome)PortableRemoteObject.narrow(obj, RuleEngineHome.class);
        return (RuleEngine)home.create();
    }
    public static String invokeRuleset(String projectName, String
rsName, String inputXML, RuleEngine ruleEngine)
    {
        String output = null;
        if(projectName == null || rsName == null || inputXML == null)
        {
            output = "Project Name or Ruleset Name or Payload should
not be NULL";
        }
        try
        {
            if(ruleEngine == null)
            {
                ruleEngine = getRuleEngine();
            }
            List xmlObjList = processPayload(inputXML);
            List opXMLObjects = ruleEngine.invokeRuleset(projectName,
rsName, xmlObjList);
            output = processReturnPayload(opXMLObjects);
        }
        catch(Exception e)

```



```

        {
            output = e.getMessage();
        }
        return output;
    }
private static String processReturnPayload(List opXMLObjects)
{
    StringBuffer op = new StringBuffer();
    int size = opXMLObjects.size();
    for(int i = 0; i < size; i++)
    {
        XMLObject obj = (XMLObject)opXMLObjects.get(i);
        String opS = obj.getXmlString();
        op.append(opS);
        if(size != 1 && i != size - 1)
        {
            op.append(ret_payloadSeparator);
            op.append(System.getProperty("line.separator"));
        }
    }
    return op.toString();
}
private static List processPayload(String inputXML)
    throws Exception
{
    List inputList = new ArrayList();
    StringReader strReader = new StringReader(inputXML);
    BufferedReader in = new BufferedReader(strReader);
    String line = null;
    StringBuffer sb = new StringBuffer();
    XMLObject xmlObj = null;
    while((line = in.readLine()) != null)
    {
        String trimmedLine = line.trim();
        if(trimmedLine.startsWith(payloadSeparator))
        {
            xmlObj =
XMLObjectFactory.createXMLObject(sb.toString());
            inputList.add(xmlObj);
        }
    }
}

```

```

        sb.setLength(0);
        ret_payloadSeparator = trimmedLine;
    } else
    {
        sb.append(line);
    }
}
if(sb.length() != 0)
{
    xmlObj = XMLObjectFactory.createXMLObject(sb.toString());
    inputList.add(xmlObj);
}
return inputList;
}
public static void main(String args[])
{
    try
    {
        processPayload(null);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
static
{
    payloadSeparator = "#";
    ret_payloadSeparator = payloadSeparator;
}
}

```

Creating the XMLHelper.java

1. In the context menu of *com.sap.helper*, choose *New > Other*
2. In the wizard that appears, choose *Class*. Choose *Next*.
3. In the screen that appears, enter *XMLHelper* in the *Name* field. Choose *Finish*.

You should see the *XMLHelper.java* window.

4. Delete all existing lines and copy the following lines into the window:

Syntax

```

package com.sap.helper;
import java.io.ByteArrayInputStream;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
public class XMLHelper {
public static String ConstructXML(String connProvider, String
destCountry, String originCountry, String typeOfLine){
    String xmlString = "<?xml version='1.0' encoding='UTF-8'?> "
+ "<CallCharges>"
+ "<Charges>" +
"<ConnectionProvider>" + connProvider + "</ConnectionProvider>" +
"<ConnectionRate>0.0</ConnectionRate>" +
"<DestinationCountry>" + destCountry + "</DestinationCountry>" +
"<OriginCountry>" + originCountry + "</OriginCountry> " +
"<TypeOfLine>" + typeOfLine + "</TypeOfLine>" +
"</Charges>" +
"</CallCharges>";
return xmlString;
}

public static String GetConnectionrate(String output)throws Exception
{
    try {
        DocumentBuilderFactory factory = DocumentBuilderFactory
            .newInstance();
        // Use the factory to create a builder
        DocumentBuilder builder = factory.newDocumentBuilder();
        ByteArrayInputStream stream = new ByteArrayInputStream(output
            .getBytes());
        Document doc = builder.parse(stream);
        // Get a list of all elements in the document.
        // Get the value of the 3rd attribute which is the connection
        charge.
        String attValue = doc.getElementsByTagName("*").item(3)
            .getTextContent();
        return attValue;
    } catch (Exception e) {

```

```

        String error = "Unable to get connection rate. Failed
Parsing"+"\\n"+output+"\\n";
        error += e.getMessage();
        throw new Exception(error,e);
    }
}

}

```

Creating the CallCharges.jsp

1. Expand the web module: *rates* node and in the context menu of the *Web Content* node, choose *New > Other*
2. In the dialog box that appears, expand the *Web* node and choose *JSP*. Choose *Next*
3. In the screen that appears, enter *CallCharges* in the *File name* field.
4. Choose *Finish*.

You should see the *CallCharges.jsp* window with the following lines:

Syntax

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
</body>
</html>

```

5. Replace *<title>Insert title here </title>* with *<title>Call Charge Calculator </title>*.
6. Copy the following lines after *<body>*

Syntax

```

<form name="ui" method="POST" action="invoker.jsp">
<h3>BRMS Invocation Client</h3>
<%
    String connProvider = (String) request.getAttribute("CONN_PROVIDER");
    String destCountry = (String) request.getAttribute("DEST_COUNTRY");
    String originCountry = (String)
request.getAttribute("ORIGIN_COUNTRY");

```

```

String typeOfLine = (String) request.getAttribute("TYPE_OF_LINE");
String callCharge = (String) request.getAttribute("CALL_CHARGE");

if(connProvider == null){
    connProvider = "";
}
if(destCountry == null){
    destCountry = "";
}
if(originCountry == null){
    originCountry = "";
}
if(typeOfLine == null){
    typeOfLine = "";
}
if(callCharge == null)
{
    callCharge = "";
}
%>
<table cellpadding="0" cellspacing="0" border="0">

<tr>
<td>Connection Provider: </td>
<td><input type="text" name="CONN_PROVIDER"
        value="<%=connProvider %>"></input></td>
</tr>
<tr>
<td>Destination Country:</td>
<td><input name="DEST_COUNTRY" type="text"
        value="<%=destCountry %>"></input></td>
</tr>
<tr>
<td>Origin Country:</td>
<td><input name="ORIGIN_COUNTRY" type="text"
        value="<%=originCountry %>"></input></td> </tr>
<tr>
<td>Type Of Line:</td>
<td><input name="TYPE_OF_LINE" type="text"

```

```

        value="<%=typeOfLine %>"></input></td>
    </tr>

    <tr>
    <td> </td>
    <td> </td>
    </tr>
    <tr>
    <td> </td>
    <td><input name="INVOKE" type="submit" value="Submit"/></td>
    </tr>
</table>
<br/>
<strong>Call Charges:</strong><br/>
<textarea id="CallCharge" name="CALL_CHARGE"
    style="width: 503px; height: 50px;"><%=callCharge %></textarea>
</form>

```

7. Save the changes.

Creating the index.jsp

1. Expand the web module: *rates* node and in the context menu of the *Web Content* node, choose *New > Other*
2. In the dialog box that appears, expand the *Web* node and choose *JSP*. Choose *Next*
3. In the screen that appears, enter *index* in the *File name* field. Choose *Next*.
4. Choose *Finish*.

You should see the *index.jsp* window with the following lines:

Syntax

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert the title here</title>
</head>
<body>
</body>
</html>

```

5. Replace `<title>Insert title here </title>` with `<title>Call Charge Calculator </title>`.
6. Copy the following line after `<body>`:

Syntax

```
<jsp:forward page="CallCharges.jsp" />
```

7. Save the changes.

Creating the invoker.jsp

1. Expand the web module: *rates* node and in the context menu of the *Web Content* node, choose *New > Other*
2. In the dialog box that appears, expand the *Web* node and choose *JSP*. Choose *Next*
3. In the screen that appears, enter *invoker* in the *File name* field. Choose *Next*.
4. Choose *Finish*.

You should see the *invoker.jsp* window with the following lines:

Syntax

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert the title here</title>
</head>
<body>
</body>
</html>
```

5. Replace `<title>Insert title here </title>` with `<title>Call Charge Calculator </title>`.
6. Copy the following lines before `<html>`

Syntax

```
<%@page import="com.sap.helper.*"%>
<%@page import="javax.xml.parsers.*" %>
<%@page import="org.w3c.dom.*"%>
<%@page import="java.io.ByteArrayInputStream"%>
```

7. Copy the following line after `<body>`:

Syntax

```

<%
String connProvider = (String) request.getParameter("CONN_PROVIDER");
String destCountry = (String) request.getParameter("DEST_COUNTRY");
String originCountry = (String)
request.getParameter("ORIGIN_COUNTRY");
String typeOfLine = (String) request.getParameter("TYPE_OF_LINE");
    // create a xml string by making use of the values inputted by the
    user.
String xmlString =
XMLHelper.ConstructXML(connProvider,destCountry,originCountry,typeOfLine);

    System.out.println(connProvider + "@@ " + destCountry+"
@@"+originCountry + " @@" + typeOfLine);
        // set project name and ruleset name.
String projectName = "demo.sap.com~ratecalculation";
//String rulesetName = "CalculateRates";
String rulesetName = "ratecalcrules";
        //invoking the rule engine
String output = EngineInvoker.invokeRuleset(projectName,
rulesetName, xmlString, null);
    // Create a factory
    //DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
    // Use the factory to create a builder
//DocumentBuilder builder = factory.newDocumentBuilder();
//ByteArrayInputStream stream = new
ByteArrayInputStream(output.getBytes());
//Document doc = builder.parse(stream);
// Get a list of all elements in the document.
// Get the value of the 3rd attribute which is the connection
charge.
String attValue = XMLHelper.GetConnectionrate(output);
    System.out.println("##### "+ output);
request.setAttribute("CONN_PROVIDER", connProvider);
request.setAttribute("DEST_COUNTRY", destCountry);
request.setAttribute("ORIGIN_COUNTRY", originCountry);
request.setAttribute("TYPE_OF_LINE", typeOfLine);
request.setAttribute("CALL_CHARGE", attValue);
%>
<jsp:forward page="CallCharges.jsp"/>

```

8. Save the changes.

Creating the Enterprise Application

Make sure that you are in the *Java EE* perspective.

1. In the SAP NetWeaver Developer Studio, choose *File > New > Project*.
2. In the wizard that appears, expand the *Development Infrastructure* node and choose *Development Component*. Choose *Next*.
3. In the screen that appears, expand the *J2EE* node and choose *Enterprise Application*. Choose *Next*.
4. In the screen that appears, choose the software component where you want to create the DCs. For example expand the '*Local Development*' node and choose *MyComponents [demo.sap.com]*. Choose *Next*.
5. In the screen that appears, enter `ratesear` in the *Name* field. Choose *Next*.
6. Choose *Next*.
7. In the *New EAR Project* screen select the *LocalDevelopment~LocalDevelopment~ rates ~demo.sap.com* checkbox. Choose *Finish*.

In the *Project Explorer* view, you should see the *ratesear* node.

Adding Dependency to the Enterprise Application

Make sure that you are in the *Development Infrastructure* perspective.

1. In the *Component Browser* view, expand the *MyComponents[demo.sap.com]* node and choose the *ratesear* node.
2. In the *Component Properties* view, choose the *Dependencies* tab.
3. Choose the *Add* button and in the wizard that appears, expand the *BRMS-FACADE[sap.com]* node and select the *tc/brms/facade* checkbox. Choose *Next*.
4. In the screen that appears, select the *Design Time*, *Deploy Time*, *Run Time* checkboxes. Choose *Finish*.

Note

- In the context menu of the *rates* and *ratesear* nodes, choose *Sync/Create Project > Sync Used DCs*
- In the dialog box that appears, choose *OK*.

Creating application.xml

Make sure that you are in the *Java EE* perspective.

1. Expand the enterprise application: *ratesear* node and in the context menu of the *Deployment Descriptor: LocalDevelopment~LocalDevelopment~ratesear~demo.sap.com* node, choose *create application.xml*. Press `Ctrl+Shift+F`

and save the changes.

Note

Expand the enterprise application: *ratesear* node, *META-INF* node double click the *application.xml* node.

You should see the *application.xml* window with the following lines:

Syntax

```

<?xml version = "1.0" encoding = "ASCII"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns="http://java.sun.com/xml/ns/javaee"

            xmlns:application="http://java.sun.com/xml/ns/javaee/application_5.xsd"
                xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/n          s/javaee/application_5.xsd"
            version="5">

<display-
name>LocalDevelopment~LocalDevelopment~ratesear~demo.sap.com</display-
name>

<module>
    <web>
        <web-uri>demo.sap.com~rates.war</web-uri>
        <context-
root>LocalDevelopment~LocalDevelopment~rates~demo.sap.com</context-
root>
    </web>
</module>
</application>

```

2. Replace `<context-root>LocalDevelopment~LocalDevelopment~buyer~demo.sap.com</context-root>` with `<context-root>CallCharges</context-root>`.

Note: Instead of `LocalDevelopment~LocalDevelopment~ buyer_wm ~demo.sap.com`, you need to enter the customized application name i.e in this tutorial, the name of the application is `CallCharges`.

Building and Deploying

Make sure you are in the *Development Infrastructure* perspective.

1. In the *Component Browser* view, expand the *Local Development, MyComponents[demo.sap.com]* nodes and in the context menu of the *rates* and *ratesear* nodes, choose *Build*.
2. In the dialog box that appears, choose *OK*.
3. In the context menu of the *ratesear* node, choose *Deploy*.
4. In the dialog box that appears, choose *OK*.
5. Open the *Infrastructure Console*, to check if the build and deploy actions have happened successfully.

Note

You can also build and deploy *rates* and *ratesear* in the Java EE perspective.

1. In the *Project Explorer* view, in the context menu of the *rates* and *ratesear* nodes, choose *Development Component > Build*.
2. In the dialog box that appears, choose *OK*.
3. In the context menu of the *ratesear* node, choose *Development Component > Build*.
4. In the dialog box that appears, choose *OK*.

Running the Web Module

1. Open the browser and enter the Application Server Address followed by the port number and the application name: `CallCharges`.
2. Enter relevant data in all available fields and choose *Submit*.

The rules get executed and you should see the call charges based on the data you entered.

Example

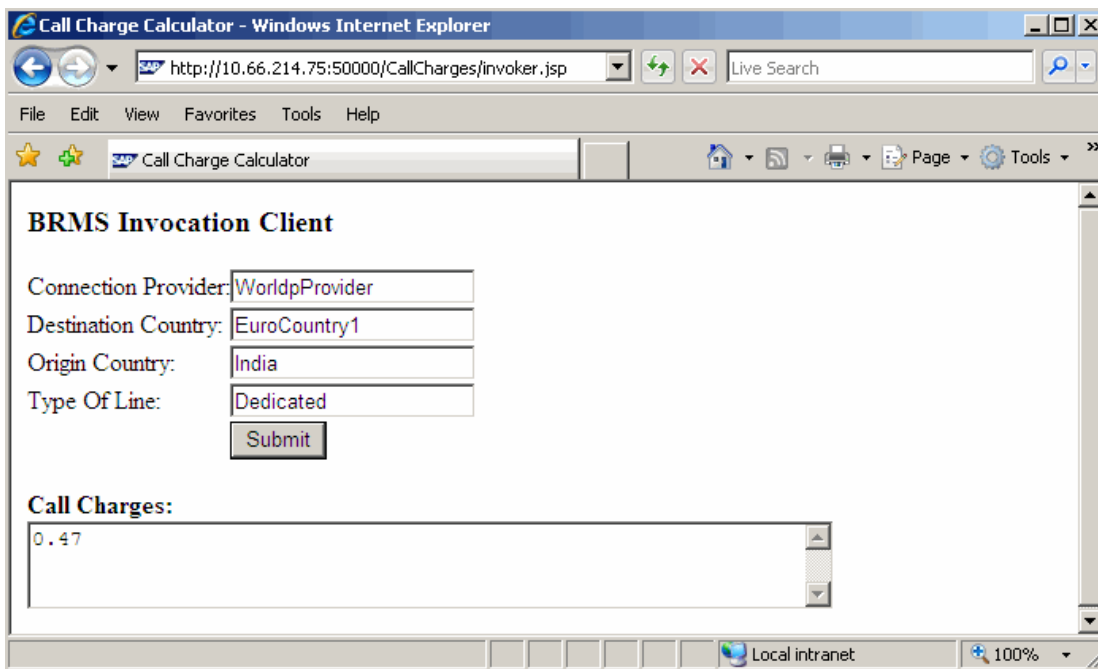
1. Enter the following data in the respective fields:

<i>Field</i>	<i>User Entry</i>
Connection Provider	WorldpProvider
Destination Country	EuroCountry1
Origin Country	India
Type Of Line	Dedicated

2. Choose *Submit*.

You should get the Call Charges as 0.47 because in the ratestable decision table, if the *Connection Provider* is WorldpProvider, *Destination Country* is EuroCountry1 and *Type Of Line* is Dedicated then the call charge is 0.47.

Here is the snapshot of the web module:



The screenshot shows a web browser window titled "Call Charge Calculator - Windows Internet Explorer". The address bar shows the URL `http://10.66.214.75:50000/CallCharges/invoke.jsp`. The browser interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with navigation and utility icons. The main content area displays the "BRMS Invocation Client" form. The form has four input fields: "Connection Provider" with the value "WorldpProvider", "Destination Country" with "EuroCountry1", "Origin Country" with "India", and "Type Of Line" with "Dedicated". A "Submit" button is located below these fields. At the bottom of the form, there is a section labeled "Call Charges:" with a text box containing the value "0.47". The browser's status bar at the bottom indicates "Local intranet" and a zoom level of "100%".

Copyright

© 2008 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, POWER5+, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.