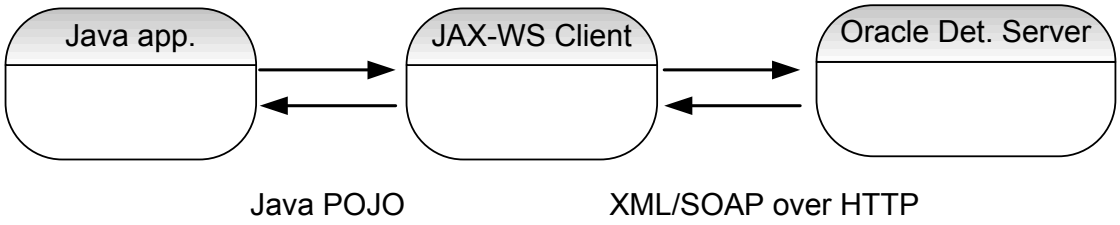
Tutorial: Creating and using a JAX-WS web service client for Oracle Determinations Server

Author: Frank Hampshire
Last Updated: 20 November 2009

Introduction

The advantage of using a rigorous and well defined interface such as a WS-I compliant web service, is the ability to integrate it with other applications, using tools to aid that integration. Using HTTP as the communication layer and XML as the request and response formats, gives a systems integrator an industry standard way of communicating, with the choice of many tools to build that integration.

This tutorial is a quick walkthrough of the direct integration of the Determinations Server running a rulebase with a simple Java application. The generated JAX-WS client will perform the job of creating the Request as a web service call, making the call, and interpreting the response.



A JAX-WS client allows you to build a web service call with Plain Old Java Objects (POJOs). The client performs the conversion to XML and a SOAP Request. The response is likewise converted from XML back to POJOs for ease of integration.

This tutorial uses the JAX-WS Reference Implementation project available from http://jax-ws.dev.java.net.

Requirements to follow the tutorial

This tutorial discusses programming in Java. You should be familiar with java, and be able to set up and run a program in an IDE of your choice (Eclipse or Netbeans for example).

The following Software is required to follow this tutorial.

- Oracle Policy Modeling 10.0
- The SimpleBenefits rulebase
- JAX-WS Reference Implementation Project (http://jax-ws.dev.java.net/)
- A Java development environment/IDE with the ability to compile and run Java 1.5 (or later) code

You should follow the instructions for installing JAX-WS correctly on your computer. This tutorial was written using version 2.1.7 of the JAX-WS RI, however older and new versions should work.

1 Compile and run the SimpleBenefits rulebase

The first thing to do is to have a look at the SimpleBenefits rulebase in Oracle Policy Modeling. This rulebase is a very simple example. It determines 3 goals.

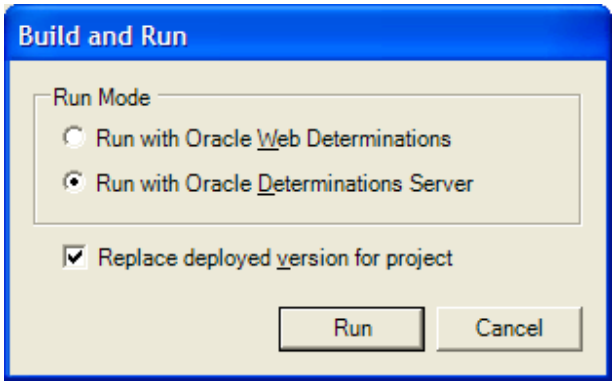
```
Is the claimant eligible for the low income allowance?
What is the claimant's low income allowance amount?
Is the claimant eligible for the teenage child allowance?
```

These goals are attributes of the global entity, and there are also child entities - the claimant's children. Eligibility for the low income allowance is based on the information on the global attribute only. Eligibility for the teenage child allowance is based on the claimant's children and their age.

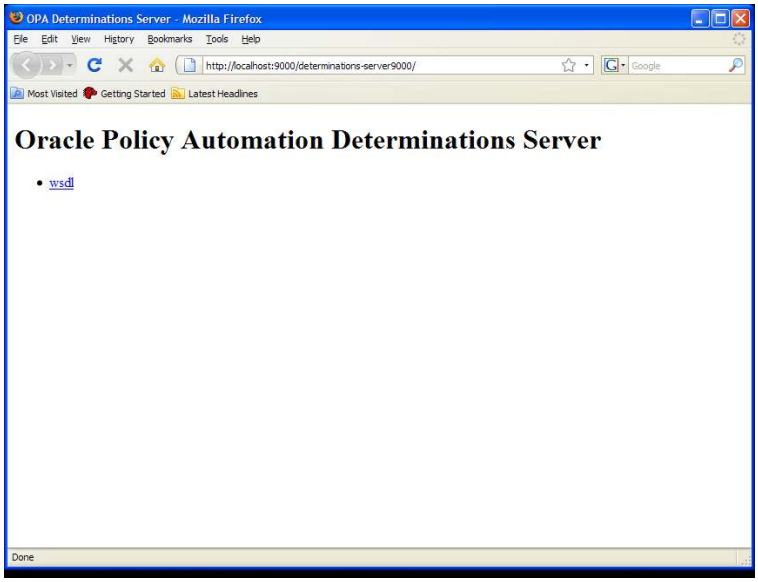
You can run this rulebase in the Determinations Server from OPM by the following method:

1. From the Build menu, choose *Build and Run*

2. When the Build and Run Dialog appears, choose Run with Oracle Determinations Server. Also select the Replace deployed version for project

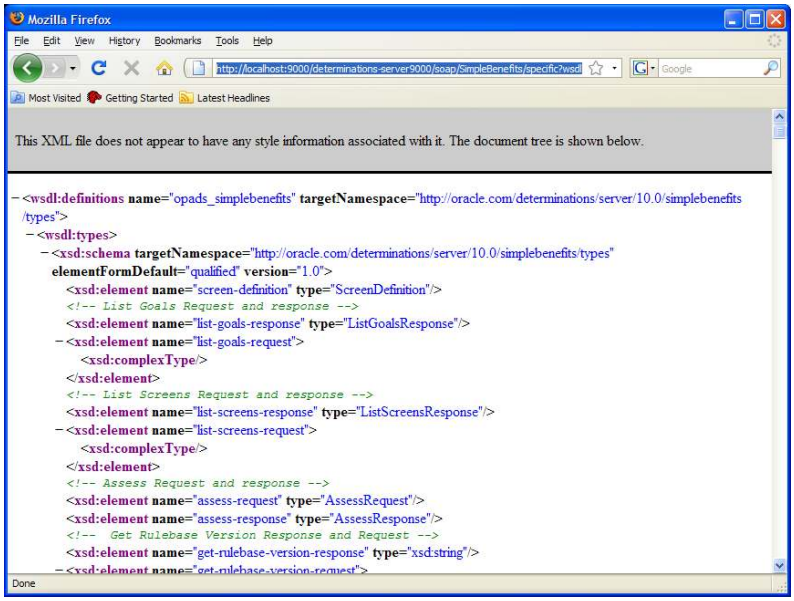


3. After a brief pause, OPM should start up your default web browser with the default page for the Determinations Server.



4. You can view the WSDLs for the rulebase by typing in the URL for the specific WSDL this will be: `http://<determinations-server-url>/soap/<rulebasename>/specific?wsdl`. In the case of OPM running SimpleBenefits, the URL should be: `http://localhost:9000/determinations-server9000/soap/SimpleBenefits/specific?wsdl`.

   If the WSDL appears as XML in the browser, then the rulebase has successfully deployed to the Determinations Server (see below).



5. The WSDL is used by the JAX-WS tool, `wsimport` to generate a Java client. You can save this page as an xml file so that the Determinations Server does not have to be running when you want to generate your client. Save the current page as `SimpleBenefits_Specific.wsdl`.

2 Compile the client

The JAX-WS `wsimport` tool can be found in the bin directory of the JAX-WS install. It is called (`wsimport.bat` for Windows and `wsimport.sh` for Unix). For detailed instructions on running this tool, see https://jax-ws.dev.java.net/jax-ws-21-ea2/docs/wsimport.html

Using `wsimport.bat` to create a Java client on Windows.

1. From a command prompt go to the `jaxws-ri\bin` directory.
2. Run the following command (all one line):

```
wsimport -s C:\SimpleBenefits\wsdls\specific\jaxws\javaclient\src
-d C:\SimpleBenefits\wsdls\specific\jaxws\javaclient\classes
C:\SimpleBenefits\wsdls\specific\SimpleBenefits_specific.wsdl
```

The arguments passed to the `wsimport` program are as follows

"`-s C:\SimpleBenefits\wsdls\specific\jaxws\javaclient\src`" specifies the output directory for the generated source files. It is very useful to have the .java files that are compiled to create the web service client.

" `-d C:\SimpleBenefits\wsdls\specific\jaxws\javaclient\classes"` specifies the output location for the compiled classes.

`"C:\SimpleBenefits\wsdls\specific\SimpleBenefits_specific.wsdl` " the last argument is the WSDL to generate the client for

3 Write a program to use the Client

Now that we have a Java client that runs against the specific wsdl for SimpleBenefits, we can create a program that uses it. The program in this tutorial is a simple command line application that creates an assess request, runs it against a Determinations Server and prints the results.

The program will take in a single optional argument, the end point to send the request to. If no end point is provided, it will default to:
`http://localhost:9000/determinations-server9000/soap/simplebenefits/specific`

The entire class `SimpleBenefitsSpecificAssessJaxWs` is provided in Appendix 2 at the end of this document.

There are several steps to creating a correct assess request using the JAX-WS generated client

3.1 Add the JAX-WS libraries to the classpath.

In order to compile and run the program, you will need to add all the jars found in the jaxws-ri\lib directory

3.2 Import the generated java client code

In order to use the JAX-WS generated code, we need to import it into our class.

```
import com.oracle.determinations.server._10_0.simplebenefits.types.*;
```

3.3 Create the assess and the entities that you will need for the request needs

In the code below, we create the assess request, the session and the entities that we intend to use (Session, Global, ListChild – for holding children entities). All these objects exist within the SimpleBenefits rulebase and also exist as XML elements within the service request that we will send. The generated objects are named for the entity and attribute names, which makes them easy to write code for.

If this were a real application, it would probably collect information on the claimant and the children from a user interface, or perhaps load them from a database. To keep things simple, we will just hard-code the values for the claimant and his/her two children.

Each entity instance needs a unique id to identify it so we will set them to "global", "child1"and "child2" respectively. If the data was coming from a database, we would probably use primary keys for the ids of the entities.

```
AssessRequest request = new AssessRequest();
Session session = new Session();
request.setSimplebenefits(session);

ListChild children = new ListChild();
session.setListChild(children);

Global global = new Global();
global.setId("global");
session.setGlobal(global);

Child child1 = new Child();
child1.setId("child1");
Child child2 = new Child();
child2.setId("child2");

children.getChild().add(child1);
children.getChild().add(child2);
```

3.4 Specify the outcomes (answers) that we want the Determinations Server to answer

Before we send the request we need to add the outcomes that we want the Determinations Server to return. In this case, there are three outcomes that we want: is the claimant eligible for the low income allowance, their low income allowance payment and is the claimant eligible for the teenage child allowance.

To request outcomes for these 3 attributes, we add each attribute, and, instead of providing a value, we set the outcome style. This indicates that instead of providing a value we are asking for the Determinations Server to return the value.

In this case we are only interested in the answer, so the outcome style will be *value-only*.

```
global.setEligibleLowIncomeAllowance(new AttributeBoolean());
global.getEligibleLowIncomeAllowance()
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

global.setLowIncomeAllowancePayment(new AttributeCurrency());
global.getLowIncomeAllowancePayment()
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

global.setEligibleTeenageAllowance(new AttributeBoolean());
global.getEligibleTeenageAllowance()
```

```
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);
```

3.5 Set attributes and relationships of the entities

Now we will set the values that we know: the claimant's income, whether the claimant a public housing client, and the ages of the children. These are attributes in the Rulebase, and also in the generated web client - attributes of the Global entity (for the claimant) and the child entity (the child's age).

```
global.setClaimantIncome(new AttributeCurrency());
global.getClaimantIncome().setNumberVal(new BigDecimal(13000.00));

global.setClaimantPublicHousingClient(new AttributeBoolean());
global.getClaimantPublicHousingClient().setBooleanVal(true);

child1.setChildAge(new AttributeNumber());
child1.getChildAge().setNumberVal(new BigDecimal(16));

child2.setChildAge(new AttributeNumber());
child2.getChildAge().setNumberVal(new BigDecimal(8));
```

Next, we need to associate the children with the claimant, via the relationship *claimantschildren*. We can create the relationship on the global, and add the two children as targets

```
global.setRelationships(new Global.Relationships());
RelationshipInstance claimantsChildren = new RelationshipInstance();
global.getRelationships().setClaimantschildren(claimantsChildren);

RelationshipTarget t1 = new RelationshipTarget();
RelationshipTarget t2 = new RelationshipTarget();
t1.setEntityId(child1);
t2.setEntityId(child2);

claimantsChildren.getTarget().add(t1);
claimantsChildren.getTarget().add(t2);
```

3.6 Call the assess method

The request is complete and ready to send. We create a specific service instance and call the assess method. This method will return an AssessResponseDocument, which should have the outcomes we asked for.

```
OpadsSimplebenefitsSpecific_Service service
    = new OpadsSimplebenefitsSpecific_Service(new URL(endPoint),
            SERVICE_QNAME);

AssessResponse response =
      service.getOpadsSimplebenefitsSpecificSOAP().assess(request);
```

The `endPoint` is the one specified when the program is run (supplied as an argument, or default and the `SERVICE_QNAME` is defined statically, at the top of the class as it never changes.

```
public static QName SERVICE_QNAME = new QName(
   "http://oracle.com/determinations/server/10.0/simplebenefits/types",
   "opads_simplebenefits_specific");
```

The Determinations Server must be running, with the SimpleBenefits rulebase deployed at the expected end point for the request to succeed.

3.7 Process the response

When the Response document is returned, we can now process it for the outcomes that were in the request. If this were a real application, the outcomes would probably be displayed to the user, or persisted to the database. In this case we will simply print them out.

```
// look for the outcomes
AttributeBoolean lowIncomeAllowance = response.getSimplebenefits()
    .getGlobal().getEligibleLowIncomeAllowance();

AttributeCurrency lowIncomeAllowancePayment =
response.getSimplebenefits()
    .getGlobal().getLowIncomeAllowancePayment();

AttributeBoolean childAllowance = response.getSimplebenefits()
    .getGlobal().getEligibleTeenageAllowance();

// print out the results
System.out.println("\n--- Results ----");

if (lowIncomeAllowance.isBooleanVal() != null) {
    System.out.println("eligible_low_income_allowance = "
        +lowIncomeAllowance.isBooleanVal());
}
else if (lowIncomeAllowance.getUnknownVal() != null) {
    System.out.println("eligible_low_income_allowance is unknown");
}
else if (lowIncomeAllowance.getUncertainVal()!= null) {
    System.out.println("eligible_low_income_allowance is uncertain");
}

if (lowIncomeAllowancePayment.getNumberVal() != null) {
    System.out.println("low_income_allowance_payment = "
        +lowIncomeAllowancePayment.getNumberVal());
}
else if (lowIncomeAllowancePayment.getUnknownVal() != null) {
    System.out.println("low_income_allowance_payment is unknown");
}
else if (lowIncomeAllowancePayment.getUncertainVal()!= null) {
    System.out.println("low_income_allowance_payment is uncertain");
```

```
}


if (childAllowance.isBooleanVal() != null) {
    System.out.println("eligible_teenage_allowance = "
        +lowIncomeAllowance.isBooleanVal());
}
else if (childAllowance.getUnknownVal() != null) {
    System.out.println("eligible_teenage_allowance is unknown");
}
else if (childAllowance.getUncertainVal()!= null) {
    System.out.println("eligible_teenage_allowance is uncertain");
}
```

3.8 Test the program

When you run the program, you should get the following output printed to standard out. From the response, we can see that all outcomes are known and that the claimant is eligible for both allowances and that the low income allowance payment is 70.0.

```
--- Starting new SimpleBenefitsAssess ---
Creating new Assess request
Setting attribute outcomes for 'eligible_low_income_allowance',
'low_income_allowance_payment' and 'eligible_teenage_allowance'
Setting claimant_income to 13000.00
Setting claimant_public_housing_client to true
Setting child_age on child1 to 16
Setting child_age on child2 to 8
Adding child1 and child2 to 'claimantschildren' relationship

--- Request Sent to Determinations Server ----

--- Response from Determinations Server ----

--- Results ----
eligible_low_income_allowance = true
low_income_allowance_payment = 70.0
eligible_teenage_allowance = true
```

4 Using a Java Client for the Generic WSDL

In the tutorial above, we created and used a Java client compiled againsr the Specific wsdl of the SimpleBenefits rulebase. The procedure for creating a client against the Generic wsdl is an identical, although the Java client will be different, and require different code to achieve the same effect.

To complete this tutorial against the Generic wsdl, you should follow the steps above but with the following differences.

4.1 Compile and run SimpleBenefits

Follow the steps outlined in *1 Compile and run SimpleBenefits*, but save the Generic wsdl instead of the specific. The url for the generic wsdl will be: `http://<determinations-server-url>/soap/<rulebasename>?wsdl`, or, in the case of this example: `http://localhost:9000/determinations-server9000/soap/SimpleBenefits?wsdl`

Save this wsdl as the file `SimpleBenefits_generic.wsdl`

4.2 Compile the client

Follow the steps outlined in *2 Compile the client,* but for the saved generic wsdl instead of the specific. So, the `wsimport` command will look like:

```
wsimport -s C:\SimpleBenefits\wsdls\generic\jaxws\javaclient\src -d
C:\SimpleBenefits\wsdls\generic\jaxws\javaclient\classes
C:\SimpleBenefits\wsdls\generic\SimpleBenefits_generic.wsdl
```

4.3 Write a program to use the Client

This is the significantly different part for the Java client generated against the generic wsdl. Although the steps are the same, the code needed to get the same result will be different for the generic java client

4.3.1 Add the JAX-WS libraries to the classpath.

As with the specific exampl, you will need to add all the jars found in the jaxws-ri\lib directory. The list of jars is as follows:

activation.jar
FastInfoset.jar
http.jar
jaxb-api.jar
jaxb-impl.jar
jaxb-xjc.jar
jaxws-api.jar
jaxws-rt.jar
jaxws-tools.jar
jsr173_api.jar
jsr181-api.jar
jsr250-api.jar
mimepull.jar
resolver.jar
saaj-api.jar
saaj-impl.jar
stax-ex.jar

streambuffer.jar
woodstox.jar

4.3.2 Import the generated java client code

The generic namespace will be different from the specific namespace. In order to use the JAX-WS generated code, we need to import it into our class

```
import com.oracle.determinations.server._10_0.rulebase.types.*;
```

4.3.3 Create the assess and the entities that you will need for the request needs

The code for creating entities is a little different for the generic client. All entities must be put into their own list (including the global entity). All the lists must have the proper entity public name as the attribute "entity-type".

From the code below, you can see that you need a few more lines to create the entity instances for the generic Java client.

```
AssessRequest request = new AssessRequest();
Session session = new Session();
request.setSessionData(session);

ListEntity listGlobal = new ListEntity();
listGlobal.setEntityType("global");
session.getListEntity().add(listGlobal);
ListEntity children = new ListEntity();
children.setEntityType("child");
session.getListEntity().add(children);

Entity global = new Entity();
global.setId("global");
listGlobal.getEntity().add(global);

Entity child1 = new Entity();
child1.setId("child1");
Entity child2 = new Entity();
child2.setId("child2");

children.getEntity().add(child1);
children.getEntity().add(child2);
```

4.3.4 Specify the outcomes (answers) that we want the Determinations Server to answer

Creating outcomes for the generic client also requires a few more lines of code. We create an AttributeOutcome for each outcome, set the attribute public name as "id" and set its outcome style.

```
AttributeOutcome lowIncomeAllowance = new AttributeOutcome();
lowIncomeAllowance.setId("eligible_low_income_allowance");
lowIncomeAllowance
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

AttributeOutcome lowIncomeAllowancePayment = new AttributeOutcome();
lowIncomeAllowancePayment.setId("low_income_allowance_payment");
lowIncomeAllowancePayment
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

AttributeOutcome teenageAllowance = new AttributeOutcome();
teenageAllowance.setId("eligible_teenage_allowance");
teenageAllowance
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

global.getAttributeOutcome().add(lowIncomeAllowance);
global.getAttributeOutcome().add(lowIncomeAllowancePayment);
global.getAttributeOutcome().add(teenageAllowance);
```

4.3.5 Set attributes and relationships of the entities

When we create attributes for the generic client we create "Attribute" objects and set the id for the Attribute to the public name of the rulebase attribute.

```
Attribute claimantIncome = new Attribute();
claimantIncome.setId("claimant_income");
claimantIncome.setNumberVal(new BigDecimal(13000.00));

Attribute claimantPHClient = new Attribute();
claimantPHClient.setId("claimant_public_housing_client");
claimantPHClient.setBooleanVal(true);

global.getAttribute().add(claimantIncome);
global.getAttribute().add(claimantPHClient);

Attribute child1Age = new Attribute();
child1Age.setId("child_age");
child1Age.setNumberVal(new BigDecimal(16));
child1.getAttribute().add(child1Age);

Attribute child2Age =  new Attribute();
child2Age.setId("child_age");
child2Age.setNumberVal(new BigDecimal(8));
child2.getAttribute().add(child2Age);
```

Relationships of entities have to be identified in the same way. When we create a Relationship, set the name to the public name of the relationship.

```
// add the children as targets of "claimants children"
Relationship claimantsChildren = new Relationship();
```

```
global.setRelationships(new ListRelationships());
global.getRelationships().getRelationship().add(claimantsChildren);
claimantsChildren.setName("claimantschildren");
RelationshipTarget t1 = new RelationshipTarget();
RelationshipTarget t2 = new RelationshipTarget();
t1.setEntityId(child1);
t2.setEntityId(child2);

claimantsChildren.getTarget().add(t1);
claimantsChildren.getTarget().add(t2);
```

4.3.6 Call the assess method

Calling the assess method in the generic client is almost identical to the specific method, although the names are different.

```
HdsRulebase service = new HdsRulebase(new URL(endPoint),
    SERVICE_QNAME);
AssessResponse response = service.getOpadsRulebaseSOAP()
    .assess(request);
```

4.3.7 Process the response

Once the response has been returned by the rulebase, we need to process the response to get the answers to the questions that we asked. This requires a little more code in the generic format because the generic XML does not distinguish between different types of entities, and it does not have specific names for the attributes we need to get.

However, by adding some simple methods to look for the attributes and entities that we need, we can simplify the code.

For details on the very simple methods `getEntityInstance` and `getAttribute`, see the full listing of the code in the Appendix below

```
// print out the results
System.out.println("\n--- Results ----");
if (lowIncomeAllowanceResp.isBooleanVal() != null) {
    System.out.println("eligible_low_income_allowance = "
        +lowIncomeAllowanceResp.isBooleanVal());
}
else if (lowIncomeAllowanceResp.getUnknownVal() != null) {
    System.out.println("eligible_low_income_allowance is unknown");
}
else if (lowIncomeAllowanceResp.getUncertainVal() != null) {
    System.out.println("eligible_low_income_allowance is uncertain");
}

if (teenageAllowanceResp.isBooleanVal() != null) {
    System.out.println("eligible_teenage_allowance = "
```

```
            +teenageAllowanceResp.isBooleanVal());
    }
    else if (teenageAllowanceResp.getUnknownVal() != null) {
        System.out.println("eligible_teenage_allowance is unknown");
    }
    else if (teenageAllowanceResp.getUncertainVal() != null) {
        System.out.println("eligible_teenage_allowance is uncertain");
    }

    if (lowIncomeAllowancePaymentResp.getNumberVal() != null) {
        System.out.println("low_income_allowance_payment = "
            +lowIncomeAllowancePaymentResp.getNumberVal());
    }
    else if (lowIncomeAllowancePaymentResp.getUnknownVal() != null) {
        System.out.println("low_income_allowance_payment is unknown");
    }
    else if (lowIncomeAllowancePaymentResp.getUncertainVal() != null) {
        System.out.println("low_income_allowance_payment is uncertain");
    }
```

4.4 Test the program

When you run the generic version of this simple program, you should get the following output printed to standard out. From the response, the results are exactly the same as the specific program.

```
--- Starting new SimpleBenefitsAssess (Generic) ---
Creating new Assess request
Setting attribute outcomes for 'eligible_low_income_allowance',
'low_income_allowance_payment' and 'eligible_teenage_allowance'
Setting claimant_income to 13000.00
Setting claimant_public_housing_client to true
Setting child_age on child1 to 16
Setting child_age on child2 to 8
Adding child1 and child2 to 'claimantschildren' relationship

--- Request Sent to Determinations Server ----

--- Response from Determinations Server ----

--- Results ----
eligible_low_income_allowance = true
eligible_teenage_allowance = true
low_income_allowance_payment = 70.0
```

5 Generic vs. Specific WSDL

As you can see for both examples you can follow the same steps to generate and use an JAX-WS Java client for a rulebase deployed on the Determinations Server. You can use either client to achieve the desired operation.

The major difference between the specific and the generic client is ease of use versus maintainability. The specific format is easier to use and much less prone to error, because attributes and relationships have specific names within the rulebase. You cannot accidently misname at attribute or a relationship using the specific format. The disadvantage with the Specific format is that adding, removing or renaming attributes and relationships will require you to regenerate the Java client using the JAX-WS wsimport tool.

The interface generated for the generic client however, can be used for any rulebase, and never needs to be recompiled. Its disadvantage is that it is easier to make mistakes with the names of attributes and relationships and the code is somewhat more cumbersome.

6 Appendix 1 – Glossary of terms

End Point – An address that can be used to communicate with a web service. For this Tutorial the end point is the location of the SimpleBenefits rulebase when deployed to the Oracle Determinations Server.

JAX-WS – Java API for XML-Based Web Services. A standard defined by JSR 224 (http://jcp.org/en/jsr/detail?id=224). This tutorial uses the JAX-WS Reference implementation to generate a Web Service client.

Oracle Determinations Server – A web application which provides Oracle Policy Automation services as a web service.

Rulebase – A compiled rule project authored in Oracle Policy Modeling.

URL – Uniform Resource Locator. A global address for documents and services on the World Wide Web

Web Service – A service provided over the Web. Typically using XML and SOAP.

WSDL – Web Service Description Language. A standard way of describing Web Services that user XML and SOAP

7 Appendix 2 - SimpleBenefitsSpecificAssess Class

```
import java.math.BigDecimal;
import java.net.URL;
import javax.xml.namespace.QName;

import com.oracle.determinations.server._10_0.simplebenefits.types.*;

public class SimpleBenefitsSpecificAssessJaxWs
{
    public static String DEFAULT_ENDPOINT
```

```
    = "http://localhost:9000/determinations-server9000/soap/simplebenefits/specific";

public static QName SERVICE_QNAME = new QName(
    "http://oracle.com/determinations/server/10.0/simplebenefits/types",
    "opads_simplebenefits_specific");

public static void main(String[] args) {

    String endPoint = args.length > 0 ? args[0] : DEFAULT_ENDPOINT;

    try {
        System.out.println("--- Starting new SimpleBenefitsAssess ---");

        System.out.println("Creating new Assess request");

        AssessRequest request = new AssessRequest();
        Session session = new Session();
        request.setSimplebenefits(session);

        ListChild children = new ListChild();
        session.setListChild(children);

        Global global = new Global();
        global.setId("global");
        session.setGlobal(global);

        Child child1 = new Child();
        child1.setId("child1");
        Child child2 = new Child();
        child2.setId("child2");

        children.getChild().add(child1);
        children.getChild().add(child2);

        System.out.println("Setting attribute outcomes for "
                +"'eligible_low_income_allowance',"
                +" 'low_income_allowance_payment'"
                +" and 'eligible_teenage_allowance'");
```

```
global.setEligibleLowIncomeAllowance(new AttributeBoolean());
global.getEligibleLowIncomeAllowance()
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

global.setLowIncomeAllowancePayment(new AttributeCurrency());
global.getLowIncomeAllowancePayment()
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

global.setEligibleTeenageAllowance(new AttributeBoolean());
global.getEligibleTeenageAllowance()
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);


System.out.println("Setting claimant_income to 13000.00");
global.setClaimantIncome(new AttributeCurrency());
global.getClaimantIncome().setNumberVal(new BigDecimal(13000.00));

System.out.println("Setting claimant_public_housing_client to true");
global.setClaimantPublicHousingClient(new AttributeBoolean());
global.getClaimantPublicHousingClient().setBooleanVal(true);

System.out.println("Setting child_age on child1 to 16");
child1.setChildAge(new AttributeNumber());
child1.getChildAge().setNumberVal(new BigDecimal(16));

System.out.println("Setting child_age on child2 to 8");
child2.setChildAge(new AttributeNumber());
child2.getChildAge().setNumberVal(new BigDecimal(8));

// add the children as targets of "claimants children"
System.out.println("Adding child1 and child2 to 'claimantschildren' relationship");
global.setRelationships(new Global.Relationships());
RelationshipInstance claimantsChildren = new RelationshipInstance();
global.getRelationships().setClaimantschildren(claimantsChildren);

RelationshipTarget t1 = new RelationshipTarget();
```

```
    RelationshipTarget t2 = new RelationshipTarget();
    t1.setEntityId(child1);
    t2.setEntityId(child2);

    claimantsChildren.getTarget().add(t1);
    claimantsChildren.getTarget().add(t2);

    OpadsSimplebenefitsSpecific_Service service
        = new OpadsSimplebenefitsSpecific_Service(new URL(endPoint),
                SERVICE_QNAME);

    System.out.println("\n--- Request Sent to Determinations Server ----");
    AssessResponse response = service.getOpadsSimplebenefitsSpecificSOAP().assess(request);

    System.out.println("\n--- Response from Determinations Server ----");


// look for the outcomes
AttributeBoolean lowIncomeAllowance = response.getSimplebenefits()
    .getGlobal().getEligibleLowIncomeAllowance();

AttributeCurrency lowIncomeAllowancePayment = response.getSimplebenefits()
    .getGlobal().getLowIncomeAllowancePayment();

AttributeBoolean childAllowance = response.getSimplebenefits()
    .getGlobal().getEligibleTeenageAllowance();

    // print out the results
System.out.println("\n--- Results ----");
if (lowIncomeAllowance.isBooleanVal() != null) {
    System.out.println("eligible_low_income_allowance = "
        +lowIncomeAllowance.isBooleanVal());
}
else if (lowIncomeAllowance.getUnknownVal() != null) {
    System.out.println("eligible_low_income_allowance is unknown");
}
else if (lowIncomeAllowance.getUncertainVal()!= null) {
    System.out.println("eligible_low_income_allowance is uncertain");
```

```
        }

        if (lowIncomeAllowancePayment.getNumberVal() != null) {
            System.out.println("low_income_allowance_payment = "
                +lowIncomeAllowancePayment.getNumberVal());
        }
        else if (lowIncomeAllowancePayment.getUnknownVal() != null) {
            System.out.println("low_income_allowance_payment is unknown");
        }
        else if (lowIncomeAllowancePayment.getUncertainVal()!= null) {
            System.out.println("low_income_allowance_payment is uncertain");
        }


        if (childAllowance.isBooleanVal() != null) {
            System.out.println("eligible_teenage_allowance = "
                +lowIncomeAllowance.isBooleanVal());
        }
        else if (childAllowance.getUnknownVal() != null) {
            System.out.println("eligible_teenage_allowance is unknown");
        }
        else if (childAllowance.getUncertainVal()!= null) {
            System.out.println("eligible_teenage_allowance is uncertain");
        }

        }
        catch(Exception e) {
            System.out.println("An error occurred"+ e.getMessage());
            e.printStackTrace();
        }
    }
}
```

8 Appendix 3 - SimpleBenefitsGenericAssess Class

```
import java.math.BigDecimal;
import java.net.URL;
import javax.xml.namespace.QName;
import com.oracle.determinations.server._10_0.rulebase.types.*;

public class SimpleBenefitsGenericAssessJaxWs
{

    public static String DEFAULT_ENDPOINT
        = "http://localhost:9000/determinations-server9000/soap/simplebenefits";
    public static QName SERVICE_QNAME = new QName(
            "http://oracle.com/determinations/server/10.0/rulebase/types",
            "hdsRulebase");

    public static void main(String[] args) {

        String endPoint = args.length > 0 ? args[0] : DEFAULT_ENDPOINT;

        try {
            System.out.println("--- Starting new SimpleBenefitsAssess (Generic) ---");

            System.out.println("Creating new Assess request");

            AssessRequest request = new AssessRequest();
            Session session = new Session();
            request.setSessionData(session);

            ListEntity listGlobal = new ListEntity();
            listGlobal.setEntityType("global");
            session.getListEntity().add(listGlobal);
            ListEntity children = new ListEntity();
            children.setEntityType("child");
            session.getListEntity().add(children);

            Entity global = new Entity();
```

```
global.setId("global");
listGlobal.getEntity().add(global);

Entity child1 = new Entity();
child1.setId("child1");
Entity child2 = new Entity();
child2.setId("child2");

children.getEntity().add(child1);
children.getEntity().add(child2);

System.out.println("Setting attribute outcomes for "
        +"'eligible_low_income_allowance',"
        +" 'low_income_allowance_payment'"
        +" and 'eligible_teenage_allowance'");

AttributeOutcome lowIncomeAllowance = new AttributeOutcome();
lowIncomeAllowance.setId("eligible_low_income_allowance");
lowIncomeAllowance
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

AttributeOutcome lowIncomeAllowancePayment = new AttributeOutcome();
lowIncomeAllowancePayment.setId("low_income_allowance_payment");
lowIncomeAllowancePayment
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

AttributeOutcome teenageAllowance = new AttributeOutcome();
teenageAllowance.setId("eligible_teenage_allowance");
teenageAllowance
    .setOutcomeStyle(AttributeOutcomeStyleEnum.VALUE_ONLY);

global.getAttributeOutcome().add(lowIncomeAllowance);
global.getAttributeOutcome().add(lowIncomeAllowancePayment);
global.getAttributeOutcome().add(teenageAllowance);

System.out.println("Setting claimant_income to 13000.00");
Attribute claimantIncome = new Attribute();
claimantIncome.setId("claimant_income");
```

```
claimantIncome.setNumberVal(new BigDecimal(13000.00));

System.out.println("Setting claimant_public_housing_client to true");
Attribute claimantPHClient = new Attribute();
claimantPHClient.setId("claimant_public_housing_client");
claimantPHClient.setBooleanVal(true);

global.getAttribute().add(claimantIncome);
global.getAttribute().add(claimantPHClient);


System.out.println("Setting child_age on child1 to 16");
Attribute child1Age = new Attribute();
child1Age.setId("child_age");
child1Age.setNumberVal(new BigDecimal(16));
child1.getAttribute().add(child1Age);

System.out.println("Setting child_age on child2 to 8");
Attribute child2Age =  new Attribute();
child2Age.setId("child_age");
child2Age.setNumberVal(new BigDecimal(8));
child2.getAttribute().add(child2Age);

// add the children as targets of "claimants children"
System.out.println("Adding child1 and child2 to 'claimantschildren' relationship");
Relationship claimantsChildren = new Relationship();
global.setRelationships(new ListRelationships());
global.getRelationships().getRelationship().add(claimantsChildren);
claimantsChildren.setName("claimantschildren");
RelationshipTarget t1 = new RelationshipTarget();
RelationshipTarget t2 = new RelationshipTarget();
t1.setEntityId(child1);
t2.setEntityId(child2);

claimantsChildren.getTarget().add(t1);
claimantsChildren.getTarget().add(t2);

HdsRulebase service = new HdsRulebase(new URL(endPoint),
```

```
    SERVICE_QNAME);

System.out.println("\n--- Request Sent to Determinations Server ----");
AssessResponse response = service.getOpadsRulebaseSOAP()
    .assess(request);

System.out.println("\n--- Response from Determinations Server ----");


// look for the outcomes
Entity globalResp = getEntityInstance(
    response.getSessionData(), "global", "global");

Attribute lowIncomeAllowanceResp
    = getAttribute(globalResp, "eligible_low_income_allowance");

Attribute lowIncomeAllowancePaymentResp
    = getAttribute(globalResp, "low_income_allowance_payment");

Attribute teenageAllowanceResp
    = getAttribute(globalResp, "eligible_teenage_allowance");

// print out the results
System.out.println("\n--- Results ----");
if (lowIncomeAllowanceResp.isBooleanVal() != null) {
    System.out.println("eligible_low_income_allowance = "
        +lowIncomeAllowanceResp.isBooleanVal());
}
else if (lowIncomeAllowanceResp.getUnknownVal() != null) {
    System.out.println("eligible_low_income_allowance is unknown");
}
else if (lowIncomeAllowanceResp.getUncertainVal() != null) {
    System.out.println("eligible_low_income_allowance is uncertain");
}

if (teenageAllowanceResp.isBooleanVal() != null) {
    System.out.println("eligible_teenage_allowance = "
        +teenageAllowanceResp.isBooleanVal());
```

```
        }
        else if (teenageAllowanceResp.getUnknownVal() != null) {
            System.out.println("eligible_teenage_allowance is unknown");
        }
        else if (teenageAllowanceResp.getUncertainVal() != null) {
            System.out.println("eligible_teenage_allowance is uncertain");
        }

        if (lowIncomeAllowancePaymentResp.getNumberVal() != null) {
            System.out.println("low_income_allowance_payment = "
                +lowIncomeAllowancePaymentResp.getNumberVal());
        }
        else if (lowIncomeAllowancePaymentResp.getUnknownVal() != null) {
            System.out.println("low_income_allowance_payment is unknown");
        }
        else if (lowIncomeAllowancePaymentResp.getUncertainVal() != null) {
            System.out.println("low_income_allowance_payment is uncertain");
        }
    }
    catch(Exception e) {
        System.out.println("An error occurred"+ e.getMessage());
        e.printStackTrace();
    }
}


// Go through the list of entity of the session data and
// return the entity that matches the entityType and id.
private static Entity getEntityInstance(Session session, String entityType, String id) {
    Entity instance = null;
    ListEntity list = null;

    // go through the entity lists to find the correct type
    for(int i =0; i < session.getListEntity().size(); i++) {
        if (entityType.equals(session.getListEntity().get(i).getEntityType())) {
            list = session.getListEntity().get(i);
            break;
        }
```

```
        }

        // go through all the list to find the entity instance with the matching id.
        if (list != null) {
            for(int i =0; i < list.getEntity().size(); i++) {
                if (id.equals(list.getEntity().get(i).getId())) {
                    instance = list.getEntity().get(i);
                    break;
                }
            }
        }

        return instance;
    }

    // Go through the list of attributes for an entity and return the
    // attribute that matches the attributeId.
    private static Attribute getAttribute(Entity entity, String attributeId) {
        Attribute attribute = null;

        for (int i =0; i < entity.getAttribute().size(); i++) {
            if (attributeId.equals(entity.getAttribute().get(i).getId())) {
                attribute = entity.getAttribute().get(i);
                break;
            }
        }

        return attribute;
    }
}
```