Tutorial: Creating and using a C# client for Oracle Determinations Server

Author: Frank Hampshire
Last Updated: 23 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 .NET application (written in C Sharp). Integration between the application and the Web Service is handled by a client, automatically generated by Visual Studio 2008. The generated web service client will perform the job of creating the request as a web service call, making the call, and interpreting the response.

Requirements to follow the tutorial

This tutorial discusses programming in C#. You should be familiar with C#, and Visual Studio.

The following software is required to follow this tutorial.

- Oracle Policy Modeling 10.0
- The SimpleBenefits rulebase
• Visual Studio (this tutorial uses VS 2008, but other versions should also work).
• .NET SDK (which should be installed with Visual Studio). .NET versions 2.0 or later will work with this tutorial.

This tutorial uses Visual Studio to generate a Service Reference. For more information on how to add a reference to a web service in Visual Studio, see http://msdn.microsoft.com/en-us/library/bb628649.aspx

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.

2 Open Visual Studio and create a new C# Console Application
3 Add a Service Reference for the Determinations Server Specific Client

Visual Studio will create Web Service Client generated from a WSDL. The best way to do this is to do the following:

1. Make sure that the service is running. In this case, make sure that Oracle Policy Modeling is running the Determinations Server.

2. From the Project Menu, choose Add Service Reference... You can also add a Service Reference by Right Clicking on the Project in the Solution Explorer and choosing Add Service Reference...

3. In the Add Service Reference dialog box, type in the URL for the Simple Benefits specific Web Service. If it is running from Oracle Policy Modeling, this should be: http://localhost:9000/determinations-server9000/SimpleBenefits/specific?wsdl
4 Use the Service Reference

Now that we have a Reference to the SimpleBenefits rulebase, we can use it in the program. The program in this tutorial will create an assess request, run it against a Determinations Server and prints the results.

The entire class SimpleBenefitsComandLine is provided in Appendix 1 at the end of this document.

There are several steps in writing the program.

4.1 Import the Service Reference classes

In order to use the SimpleBenefits Service Reference client generated code, we need to import it into our class.

```csharp
using SimpleBenefitsDotNetApp.SimpleBenefitsSpecific;
```

4.2 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 child entities). All these objects exist within the SimpleBenefits rulebase and also exist as XML elements within the service request that we will send. We will use objects generated as part of the service reference, which match the XML of the request that we need to send./n

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. If the data was coming from a database, we would probably use primary keys for the ids of the entities, but, again, to keep things simple, we will set them to "global", "child1" and "child2" respectively.

```java
AssessRequest request = new AssessRequest();
request.simplebenefits = new Session();
request.simplebenefits.listchild = new listchild();

global g = new global();
request.simplebenefits.global = g;
request.simplebenefits.global.id = "global";

child child1 = new child();
child1.id = "child1";

child child2 = new child();
child2.id = "child2";

request.simplebenefits.listchild.child
    = new child[] { child1, child2 };```

4.3 **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?
- The claimant's low income allowance payment
- 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**.

```java
g.eligible_low_income_allowance = new AttributeBoolean();
g.eligible_low_income_allowance.outcomestyle = 'value-only';```
= AttributeOutcomeStyleEnum.valueonly;
g.eligible_low_income_allowance
.outcomestyleSpecified = true;

g.low_income_allowance_payment = new AttributeCurrency();
g.low_income_allowance_payment.outcomestyle
 = AttributeOutcomeStyleEnum.valueonly;
g.low_income_allowance_payment.outcomestyleSpecified = true;

g.eligible_teenage_allowance = new AttributeBoolean();
g.eligible_teenage_allowance.outcomestyle
 = AttributeOutcomeStyleEnum.valueonly;
g.eligible_teenage_allowance.outcomestyleSpecified = true;

Note: because the attribute "outcome-style" is an optional attribute, when we set this
attribute, we also need to indicate that we have specified the attribute by setting the
property outcomestyleSpecified to true.

4.4 Set attributes and relationships of the entities

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

g.claimant_income = new AttributeCurrency();
g.claimant_income.Item = new Decimal(13000.00);

g.claimant_public_housing_client = new AttributeBoolean();
g.claimant_public_housing_client.Item = true;

c1.child_age = new AttributeNumber();
c1.child_age.Item = new Decimal(16);

c2.child_age = new AttributeNumber();
c2.child_age.Item = new Decimal(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.

g.relationships = new globalRelationships();
g.relationships.claimantschildren = new RelationshipInstance();

RelationshipTarget t1 = new RelationshipTarget();
RelationshipTarget t2 = new RelationshipTarget();
t1.entityid = c1.id;
t2.entityid = c2.id;

g.relationships.claimantschildren.target
 = new RelationshipTarget[] { t1, t2 };}
4.5 Call the assess method

The request is complete and ready to send. We create a specific service instance and call the assess method.

We need to wrap the AssessRequest in the AssessRequest1 object (that’s just the way .NET interprets this web service). Likewise the response is returned wrapped in an AssessResponse1 object. The wrapped AssessResponse object should have the outcomes we asked for.

```csharp
opads_simplebenefits_specific service = new opads_simplebenefits_specificClient();
AssessRequest1 requestDoc = new AssessRequest1(request);
AssessResponse1 responseDoc = service.Assess(requestDoc);
AssessResponse response = responseDoc.assessresponse;
```

4.6 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.

```csharp
AttributeBoolean lowIncomeAllowance = response.simplebenefits.global.eligible_low_income_allowance;
AttributeCurrency lowIncomeAllowancePayment = response.simplebenefits.global.low_income_allowance_payment;
AttributeBoolean teenageAllowance = response.simplebenefits.global.eligible_teenage_allowance;

Console.WriteLine("\n--- Results ----");
if (lowIncomeAllowance.Item is Boolean) {
    Console.WriteLine("eligible_low_income_allowance = " + lowIncomeAllowance.Item);
} else if (lowIncomeAllowance.Item is UnknownValue) {
    Console.WriteLine("eligible_low_income_allowance is unknown");
} else if (lowIncomeAllowance.Item is UncertainValue) {
    Console.WriteLine("eligible_low_income_allowance is uncertain");
}

if (lowIncomeAllowancePayment.Item is Decimal) {
    Console.WriteLine("low_income_allowance_payment = " + lowIncomeAllowancePayment.Item);
}
```
else if (lowIncomeAllowancePayment.Item is UnknownValue)
{
    Console.WriteLine("low_income_allowance_payment is unknown");
} else if (teenageAllowance.Item is UncertainValue)
{
    Console.WriteLine("low_income_allowance_payment is uncertain");
}

if (teenageAllowance.Item is Boolean)
{
    Console.WriteLine("eligible_teenage_allowance = "+lowIncomeAllowance.Item);
} else if (teenageAllowance.Item is UnknownValue)
{
    Console.WriteLine("eligible_teenage_allowance is unknown");
} else if (teenageAllowance.Item is UncertainValue)
{
    Console.WriteLine("eligible_teenage_allowance is uncertain");
}

### 4.7 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
5 Running against a different endpoint

The endpoint of your rulebase running in the Oracle Determination Server will change when you move it to a different server, or you change some other aspect of the deployment like the port the Determinations Server runs on. For example, you may write your integration code against the Determinations Server deployed in a test environment, and, of course, you want your code to run against the production deployment when it is released.

The endpoint for a Service Reference is stored in the app.config file for the program. In Visual Studio, you should be able to see a file in your project called app.config. If you open that file, you should be able to a section starting with the `client` element.

In the endpoint element, you should be able to see details of the service endpoint, including the URL to run against. You can change this XML to match the Determinations Server that you want to run against.

You should only need to change the `address` attribute of the `endpoint` element

```xml
<client>
  <endpoint address="http://localhost:9000/determinations-server9000/soap/simplebenefits
             binding="basicHttpBinding"
             bindingConfiguration="opads_simplebenefits_generic"
             contract="SimpleBenefitsGen.opads_simplebenefits_generic"
             name="opadsRulebaseSOAP" />
</client>
```

6 Using a Service Reference for the Generic WSDL

In the tutorial above, we created and used a Web Service Reference client compiled against the Specific wsdl of the SimpleBenefits rulebase. The procedure for creating a client against the Generic wsdl is an identical one, although the Web Service Reference 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.

6.1 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:

Save this wsdl as the file SimpleBenefits_generic.wsdl

6.2 Create a Service Reference to the Generic

Follow the steps outlined in 3 Add a Service Reference for the Determinations Server Specific Client, but for the generic wsdl instead of the specific.

![Add Service Reference dialog]

6.3 Write a program to use the Service Reference

This is the significantly different part for the Reference generated against the generic service. Although the steps are the same, the code needed to get the same result will be different.

6.3.1 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

```csharp
```

6.3.2 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 service. 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 service.

```csharp
AssessRequest request = new AssessRequest();
ListEntity globalList = new ListEntity();
globalList.entitytype = "global";
ListEntity childList = new ListEntity();
childList.entitytype = "child";
request.sessiondata = new ListEntity[] { globalList, childList };

Entity reqGlobal = new Entity();
reqGlobal.id = "global";
globalList.entity = new Entity[] { reqGlobal };

Entity child1 = new Entity();
child1.id = "child1";

Entity child2 = new Entity();
child2.id = "child2";

childList.entity = new Entity[] { child1, child2 };
```

6.3.3 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.

```csharp
AttributeOutcome eligible_low_income_allowance = new AttributeOutcome();
eligible_low_income_allowance.id = "eligible_low_income_allowance";
eligible_low_income_allowance.outcomestyle = AttributeOutcomeStyleEnum.valueonly;
eligible_low_income_allowance.outcomestyleSpecified = true;

AttributeOutcome low_income_allowance_payment = new AttributeOutcome();
low_income_allowance_payment.id = "low_income_allowance_payment";
```
low_income_allowance_payment.outcome_style = AttributeOutcomeStyleEnum.valueonly;
low_income_allowance_payment.outcome_style_specified = true;

 AttributeOutcome eligible_teenage_allowance = new AttributeOutcome();
eligible_teenage_allowance.id = "eligible_teenage_allowance";
eligible_teenage_allowance.outcome_style = AttributeOutcomeStyleEnum.valueonly;
eligible_teenage_allowance.outcome_style_specified = true;

reqGlobal.attributeoutcome = new AttributeOutcome[] { eligible_low_income_allowance,
                                            low_income_allowance_payment,
                                            eligible_teenage_allowance};

6.3.4 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. We also have to specify for each attribute, what value type it will use (ItemChoiceType)

 Attribute claimant_income = new Attribute();
claimant_income.id = "claimant_income";
claimant_income.Item = new Decimal(13000);
claimant_income.ItemElementName = ItemChoiceType.numberval;

 Attribute claimant_public_housing_client = new Attribute();
claimant_public_housing_client.id = "claimant_public_housing_client";
claimant_public_housing_client.Item = true;
claimant_public_housing_client.ItemElementName = ItemChoiceType.booleanval;

reqGlobal.attribute = new Attribute[] { claimant_income, claimant_public_housing_client};

 Attribute child1_age = new Attribute();
child1_age.id = "child_age";
child1_age.Item = new Decimal(16);
child1_age.ItemElementName = ItemChoiceType.numberval;

child1.attribute = new Attribute[] { child1_age};

 Attribute child2_age = new Attribute();
child2_age.id = "child_age";
child2_age.Item = new Decimal(8);
child2_age.ItemElementName = ItemChoiceType.numberval;

child2.attribute = new Attribute[] { child2_age};

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.
Console.WriteLine("Adding child1 and child2 to 'claimantschildren' relationship");
Relationship claimantschildren = new Relationship();
claimantschildren.name = "claimantschildren";

RelationshipTarget t1 = new RelationshipTarget();
RelationshipTarget t2 = new RelationshipTarget();
t1.entityid = child1.id;
t2.entityid = child2.id;

claimantschildren.target = new RelationshipTarget[] { t1, t2 };

reqGlobal.relationships = new ListRelationships();
reqGlobal.relationships.relationship = new Relationship[] { claimantschildren };

6.3.5 Call the assess method

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

opads_simplebenefits_genericClient service = new opads_simplebenefits_genericClient();
AssessResponse response = service.Assess(request);

6.3.6 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

Entity respGlobal = GetEntityInstance(response.sessiondata, "global", "global");

Console.WriteLine("\n--- Response from Determinations Server ----");

// look for the outcomes
Attribute lowIncomeAllowance = GetAttribute(respGlobal, "eligible_low_income_allowance");
Attribute lowIncomeAllowancePayment = GetAttribute(respGlobal, "low_income_allowance_payment");
Attribute teenageAllowance = GetAttribute(respGlobal, "eligible_teenage_allowance");

// print out the results
Console.WriteLine("\n--- Results ----");

if (lowIncomeAllowance.Item is Boolean)
{
    Console.WriteLine("eligible_low_income_allowance = "+ lowIncomeAllowance.Item);
}
else if (lowIncomeAllowance.Item is UnknownValue)
{
    Console.WriteLine("eligible_low_income_allowance is unknown");
}
else if (lowIncomeAllowance.Item is UncertainValue)
{
    Console.WriteLine("eligible_low_income_allowance is uncertain");
}

if (lowIncomeAllowancePayment.Item is Decimal)
{
    Console.WriteLine("low_income_allowance_payment = "+ lowIncomeAllowancePayment.Item);
}
else if (lowIncomeAllowancePayment.Item is UnknownValue)
{
    Console.WriteLine("low_income_allowance_payment is unknown");
}
else if (teenageAllowance.Item is UncertainValue)
{
    Console.WriteLine("low_income_allowance_payment is uncertain");
}

if (teenageAllowance.Item is Boolean)
{
    Console.WriteLine("eligible_teenage_allowance = "+ lowIncomeAllowance.Item);
}
else if (teenageAllowance.Item is UnknownValue)
{
    Console.WriteLine("eligible_teenage_allowance is unknown");
}
else if (teenageAllowance.Item is UncertainValue)
{
    Console.WriteLine("eligible_teenage_allowance is uncertain");
}

6.4 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

**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 accidentally 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.
Appendix 1 – Glossary of terms

.NET – A framework provided by Microsoft for building applications. For more information on .NET see http://www.microsoft.com/net/


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.

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.

Service Reference – In Visual Studio, this is a reference to an external Service. In the context of this tutorial the service is a Web Service.

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 use XML and SOAP

Appendix 2 - SimpleBenefitsComandLine Class

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SimpleBenefitsDotNetApp.SimpleBenefitsSpecific;

namespace SimpleBenefitsDotNetApp
{
    class SimpleBenefitsComandLine
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("--- Starting new SimpleBenefitsAssess (Generic) ---");
                Console.WriteLine("Creating new Assess request");
                AssessRequest request = new AssessRequest();
                request.simplebenefits = new Session();
                request.simplebenefits.listchild = new listchild();
                global reqGlobal = new global();
                reqGlobal.id = "global";
                request.simplebenefits.global = reqGlobal;
                child child1 = new child();
                child1.id = "child1";
                child child2 = new child();
                child2.id = "child2";
                request.simplebenefits.listchild.child
                    = new child[] { child1, child2 };
                Console.WriteLine("Setting attribute outcomes for " + "'eligible_low_income_allowance'," + "'low_income_allowance_payment'"," + " and 'eligible_teenage_allowance'");
                reqGlobal.eligible_low_income_allowance = new AttributeBoolean();
            }
        }
    }
}
reqGlobal.eligible_low_income_allowance.outcomestyle = AttributeOutcomeStyleEnum.valueonly;
reqGlobal.eligible_low_income_allowance.outcomestyleSpecified = true;

reqGlobal.low_income_allowance_payment = new AttributeCurrency();
reqGlobal.low_income_allowance_payment.outcomestyle = AttributeOutcomeStyleEnum.valueonly;
reqGlobal.low_income_allowance_payment.outcomestyleSpecified = true;

reqGlobal.eligible_teenage_allowance = new AttributeBoolean();
reqGlobal.eligible_teenage_allowance.outcomestyle = AttributeOutcomeStyleEnum.valueonly;
reqGlobal.eligible_teenage_allowance.outcomestyleSpecified = true;

Console.WriteLine("Setting claimant_income to 13000.00");
reqGlobal.claimant_income = new AttributeCurrency();
reqGlobal.claimant_income.Item = new Decimal(13000.00);

Console.WriteLine("Setting claimant_public_housing_client to true");
reqGlobal.claimant_public_housing_client = new AttributeBoolean();
reqGlobal.claimant_public_housing_client.Item = true;

Console.WriteLine("Setting child_age on child1 to 16");
child1.child_age = new AttributeNumber();
child1.child_age.Item = new Decimal(16);

Console.WriteLine("Setting child_age on child2 to 8");
child2.child_age = new AttributeNumber();
child2.child_age.Item = new Decimal(8);

// add the children as targets of "claimants children"
Console.WriteLine("Adding child1 and child2 to 'claimantschildren' relationship");
reqGlobal.relationships = new globalRelationships();
reqGlobal.relationships.claimantschildren = new RelationshipInstance();

RelationshipTarget t1 = new RelationshipTarget();
RelationshipTarget t2 = new RelationshipTarget();
t1.entityid = child1.id;
t2.entityid = child2.id;

reqGlobal.relationships.claimantschildren.target
    = new RelationshipTarget[] { t1, t2 };

Console.WriteLine("\n--- Request Sent to Determinations Server ----");
opads_simplebenefits_specific service = new opads_simplebenefits_specificClient();
AssessRequest1 requestDoc = new AssessRequest1(request);
AssessResponse1 responseDoc = service.Assess(requestDoc);
AssessResponse response = responseDoc.assessresponse;

Console.WriteLine("\n--- Response from Determinations Server ----");

// look for the outcomes
AttributeBoolean lowIncomeAllowance = response.simplebenefits
    .global.eligible_low_income_allowance;
AttributeCurrency lowIncomeAllowancePayment = response.simplebenefits
    .global.low_income_allowance_payment;
AttributeBoolean teenageAllowance = response.simplebenefits
    .global.eligible_teenage_allowance;

// print out the results
Console.WriteLine("\n--- Results ----");

if (lowIncomeAllowance.Item is Boolean) {
    Console.WriteLine("eligible_low_income_allowance = "
        + lowIncomeAllowance.Item);
}
else if (lowIncomeAllowance.Item is UnknownValue) {
    Console.WriteLine("eligible_low_income_allowance is unknown");
}
else if (lowIncomeAllowance.Item is UncertainValue) {
    Console.WriteLine("eligible_low_income_allowance is uncertain");
}
if (lowIncomeAllowancePayment.Item is Decimal) {
    Console.WriteLine("low_income_allowance_payment = "
                     + lowIncomeAllowancePayment.Item);
} else if (lowIncomeAllowancePayment.Item is UnknownValue)
{
    Console.WriteLine("low_income_allowance_payment is unknown");
} else if (teenageAllowance.Item is UncertainValue)
{
    Console.WriteLine("low_income_allowance_payment is uncertain");
}

if (teenageAllowance.Item is Boolean)
{
    Console.WriteLine("eligible_teenage_allowance = "
                     + lowIncomeAllowance.Item);
} else if (teenageAllowance.Item is UnknownValue)
{
    Console.WriteLine("eligible_teenage_allowance is unknown");
} else if (teenageAllowance.Item is UncertainValue)
{
    Console.WriteLine("eligible_teenage_allowance is uncertain");
}

} catch (Exception e)
{
    Exception innerE = e.InnerException;
    Console.WriteLine("Error occurred " + e.Message);
}

} }
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SimpleBenefitsDotNetApp.SimpleBenefitsSpecific;

namespace SimpleBenefitsDotNetApp
{
    class SimpleBenefitsComandLine
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("--- Starting new SimpleBenefitsAssess (Generic) ---");
                Console.WriteLine("Creating new Assess request");
                AssessRequest request = new AssessRequest();
                request.simplebenefits = new Session();
                request.simplebenefits.listchild = new listchild();
                global reqGlobal = new global();
                reqGlobal.id = "global";
                request.simplebenefits.global = reqGlobal;
                child child1 = new child();
                child1.id = "child1";
                child child2 = new child();
            }
        }
    }
}
child2.id = "child2";

request.simplebenefits.listchild.child
= new child[] { child1, child2 };  

Console.WriteLine("Setting attribute outcomes for "
+"'eligible_low_income_allowance'," 
+"'low_income_allowance_payment'"
+" and 'eligible_teenage_allowance'");

reqGlobal.eligible_low_income_allowance = new AttributeBoolean();
reqGlobal.eligible_low_income_allowance.outcomestyle
= AttributeOutcomeStyleEnum.valueonly;
reqGlobal.eligible_low_income_allowance .outcomestyleSpecified = true;

reqGlobal.low_income_allowance_payment = new AttributeCurrency();
reqGlobal.low_income_allowance_payment.outcomestyle
= AttributeOutcomeStyleEnum.valueonly;
reqGlobal.low_income_allowance_payment.outcomestyleSpecified = true;

reqGlobal.eligible_teenage_allowance = new AttributeBoolean();
reqGlobal.eligible_teenage_allowance.outcomestyle
= AttributeOutcomeStyleEnum.valueonly;
reqGlobal.eligible_teenage_allowance.outcomestyleSpecified = true;

Console.WriteLine("Setting claimant_income to 13000.00");
reqGlobal.claimant_income = new AttributeCurrency();
reqGlobal.claimant_income.Item = new Decimal(13000.00);

Console.WriteLine("Setting claimant_public_housing_client to true");
reqGlobal.claimant_public_housing_client = new AttributeBoolean();
reqGlobal.claimant_public_housing_client.Item = true;

Console.WriteLine("Setting child_age on child1 to 16");
child1.child_age = new AttributeNumber();
child1.child_age.Item = new Decimal(16);
Console.WriteLine("Setting child age on child2 to 8");

child2.age = new AttributeNumber();
child2.age.Item = new Decimal(8);

// add the children as targets of "claimants children" relationship
reqGlobal.relationships.claimantschildren.target = new RelationshipTarget[] { t1, t2 };

Console.WriteLine("--- Request Sent to Determinations Server ----");

opads_simplebenefits_specific service = new opads_simplebenefits_specificClient();
AssessRequest1 requestDoc = new AssessRequest1(request);
AssessResponse1 responseDoc = service.Assess(requestDoc);
AssessResponse response = responseDoc.assessresponse;

Console.WriteLine("--- Response from Determinations Server ----");

// look for the outcomes
AttributeBoolean lowIncomeAllowance = response.simplebenefits.global.eligible_low_income_allowance;
AttributeCurrency lowIncomeAllowancePayment = response.simplebenefits.global.low_income_allowance_payment;
AttributeBoolean teenageAllowance = response.simplebenefits.global.eligible_teenage_allowance;

// print out the results
Console.WriteLine("--- Results ----");
if (lowIncomeAllowance.Item is Boolean) {
    Console.WriteLine("eligible_low_income_allowance = "
            +lowIncomeAllowance.Item);
} else if (lowIncomeAllowance.Item is UnknownValue) {
    Console.WriteLine("eligible_low_income_allowance is unknown");
} else if (lowIncomeAllowance.Item is UncertainValue) {
    Console.WriteLine("eligible_low_income_allowance is uncertain");
}

if (lowIncomeAllowancePayment.Item is Decimal) {
    Console.WriteLine("low_income_allowance_payment = "
            +lowIncomeAllowancePayment.Item);
} else if (lowIncomeAllowancePayment.Item is UnknownValue) {
    Console.WriteLine("low_income_allowance_payment is unknown");
} else if (teenageAllowance.Item is UncertainValue) {
    Console.WriteLine("low_income_allowance_payment is uncertain");
}

if (teenageAllowance.Item is Boolean) {
    Console.WriteLine("eligible_teenage_allowance = "
            +lowIncomeAllowance.Item);
} else if (teenageAllowance.Item is UnknownValue) {
    Console.WriteLine("eligible_teenage_allowance is unknown");
} else if (teenageAllowance.Item is UncertainValue) {
    Console.WriteLine("eligible_teenage_allowance is uncertain");
}
catch (Exception e)
    {
        Exception innerE = e.InnerException;
        Console.WriteLine("Error occurred "+ e.Message);
    }  
}