Export Ofbiz Services that use complex type parameters via SOAP using AXIS2

This page is deprecated since r892712

h1. How to export Ofbiz services that use complex type parameters (java beans, including arrays of beans) via SOAP using AXIS2.

Alfredo Rueda Unsain
http://www.opentrends.net
Note: Attachments Section includes an Axis2 Ofbiz Component Example that you may adapt to your own needs.

1. Architecture of the solution:

Axis2 Server will run as a webapp deployed inside Ofbiz Server. Then, you will deploy a web service in the Axis2 repository, using the simple Axis2 POJO system. Finally, the web service will have access to Ofbiz resources through default delegator and default dispatcher. The web service will play the rol of a thin proxy that will call the Ofbiz service that must be exported and then will do some transformation of parameters as needed (for instance: List of GenericValues to an Array of Java Beans).

2. System components

This solution has been tested with the following components:

1. Axis2 1.4.1 : http://ws.apache.org/axis2/download/1_4_1/download.cgi
2. Opentaps 1.0 : svn://svn.opentaps.org/opentaps_all/versions/1.0/trunk : rev 5910
3. Ant 1.7.0 : http://ant.apache.org/bindownload.cgi
4. Ubuntu 8.04

3. Steps to accomplish the task:

3.1 Deploy Axis2 WebApp inside Ofbiz Server

1. Unpack axis2.war (WAR Distribution) in the webapp directory of an existing Ofbiz component located in the hot-deploy directory, for instance: "crmsfa" (this is only for a quick test, the real deployment must be done with a fresh new Ofbiz component named Axis2: Attachment Section includes an example). Then, edit the file ofbiz-component.xml, located in the root of the "crmsfa" directory, and add a webapp element that will point to the Axis2 previously unpacked:

```
<webapp name="axis2"
  title="Axis 2"
  server="default-server"
  location="webapp/axis2"
  mount-point="/axis2"
  app-bar-display="false"/>
```

2. Remove commons-logging-1.1.1.jar and xercesImpl-2.8.1.jar libraries from axis2/WEB-INF/lib directory to avoid class clashing problems between Axis2 and Ofbiz libraries.
3. Restart Ofbiz server.
4. Go to http://<host:port>/axis2. It should produce the Axis2 Web Application Home Page. Now you are ready to deploy your web services in Ofbiz!

3.2 Build and Run a sample Web Service

1. First of all, download Axis2 Standard Binary Distribution from http://ws.apache.org/axis2/download/1_4_1/download.cgi. Then, we will run the pojoguide sample located at Axis2_HOME/samples/pojoguide where Axis2_HOME must point to the directory where you installed Axis2 Standard Binary Distribution. To perform this task follow the instructions at http://ws.apache.org/axis2/1_4_1/pojoguide.html. Basically, you need to perform these simple steps:

```
1. Set <property name="axis2.home" value="" /> in the file build.xml to point to the Axis2
Standard Binary  Distribution home directory
2. Run the following command at the Axis2_HOME/samples/pojoguide directory:
   ant generate.service

-- This Ant task builds the service relevant source, and copies the files to build/WeatherService.

3. Now, deploy the web service by copying the file "wheatherservice.aar"
   to "crmsfa/webapp/axis2/repository/services".

4. In order to test the web service, run the command:
   ant rpc.client

-- This task builds the client relevant files, builds a JAR
   at build/lib/rpc-client.jar, and then runs the client.
```

~~2. Now extend the sample using an array of beans. You may add an operation that returns array of weather beans. To understand all the possibilities of the Axis2 POJO System, it's recommended to check http://wso2.org/library/2893.~~
~~3. Optionally, you can play with the sample and return a bean that includes an array of beans. This is useful if you need to return some extra information, as for instance a description of the result.~~

~~3.3 Call an Ofbiz Service from your Web Service~~

~~The next step involves calling an Ofbiz service from the recently deployed Web Service.~~

~~1. Add the following code to the Java class that will implement a WebServices Facade:~~

```
import java.util.*;
import org.ofbiz.entity.GenericDelegator;
import org.ofbiz.service.GenericDispatcher;
import org.ofbiz.base.util.*;
import org.ofbiz.entity.util.*;

GenericDelegator delegator = GenericDelegator.getGenericDelegator("default");
LocalDispatcher dispatcher = GenericDispatcher.getLocalDispatcher("default",delegator);
GenericValue admin = null;
try {
        admin = delegator.findByPrimaryKey("UserLogin", UtilMisc.toMap("userLoginId", "admin"));
} catch (GenericEntityException e1) {
        e1.printStackTrace();
}
```

~~This Java class may contain a new method for each Ofbiz Service that you want to export.~~

-2. You will also need to include Ofbiz jars in the <path id="build.class.path"> element of the file "build.xml" provided with the WeatherService sample:-

```
<fileset dir="${ofbiz.home}/framework/base/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/framework/base/build/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/framework/entity/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/framework/entity/build/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/framework/security/build/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/framework/service/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/framework/service/build/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/framework/minilang/build/lib" includes="*.jar"/>
<fileset dir="${ofbiz.home}/applications/content/build/lib" includes="*.jar"/>
```

~~3. Finally you are able to call Ofbiz services from your Web Service! So the last stage involves:~~
~~3.1. Call the Ofbiz Service that you want to export~~

```
Map result = null;

Object[] params = {
"param_a", param_a,
"param_b", param_b,
"param_c", param_c,
"userLogin", admin
};

try {
     result = dispatcher.runSync("Ofbiz service",
                                 UtilMisc.toMap(params));
     List items = (List) result.get("items");
     // Example: Transform "items" from List of Generic Values to an Array of Java Beans

} catch (GenericServiceException e) {
        e.printStackTrace();
}
```

~~3.2. Adapt the Ofbiz Service Output (Output Parameters, Errors and Exceptions, etc) to an interface suitable to Web Services programming style: For instance: List of Generic Values to an Array of Java Beans~~

~~4. Feedback~~

~~This HowTo was born from~~
~~http://www.nabble.com/Array-of-pojos-as-an-output-parameter-of-an-ofbiz-service-exported-via-soap-td20601376.html~~

~~Thanks to jacques.le.roux for suggesting Axis integration approach.~~
~~Please, send feedback to ofbiz mailing list!~~

~~Alfredo Rueda Unsain~~
~~http://www.opentrends.net~~
~~Note: Attachment Section includes an Axis2 Ofbiz Component Example that you may adapt to your own needs.~~

~~This solution was developed by http://www.opentrends.net as a consulty service for http://www.upcnet.es~~

~~This interesting thread leaded to Paul Piper's axis2forofbiz.rar. I will review and maybe add to OFBiz...~~