Web Services-Controlling WSDL Service Location

This documentation was written by Andrew Lindesay (http://www.lindesay.co.nz) in 2006 as part of supported code in the LEWOStuff framework, but this material has been transcribed here. It was written around the time of WebObjects 5.2 and 5.3 on the 1.4 JVM.

AXIS provides a number of handlers for ultimately executing the WS. This generally works in the WebObjects environment out of the box. However, in the case of a servlet deployment, the URLs are manipulated by the servlet container so they look something like this.

```
...ects/FOO.woa/ws/FooService;jsessionid=abc.i1?wsdl
```

If a client starts a session on a WS invocation then you want all subsequent requests to return to this same servlet container (Here we assume a redundant deployment where there are a number of servlet containers.) where the session resides. However the servlet-container's HTTP adaptor typically only knows this by looking at the modified URL. Unfortunately, the default WSDL generated by AXIS would have a section like this.

```
...
<wsdl:service name="FooService">
  <wsdl:port name="FooService" binding="impl:FooService SoapBinding">
    <wsdlsoap:address location="http://foo.co.nz/FOO/WebObjects/FOO.woa/ws/FooService"/>
  </wsdl:port>
</wsdl:service>
...
```

A WSDL like this does not contain the modified URL caused by the context having a session in the location for the service. The example handler *LEWOWebServicesWSDLLocationHandler* below which is supplied in LEWOStuff will correct the location supplied in the WSDL to contain the session information. To install this handler for use in WebObjects, locate the *server.wsdd* file within your project and modify the following section so that it appears as follows.

```
...
<transport name="http">
  <requestFlow>
    <handler type="HTTPActionHandler"/>
    <handler type="URLMapper"/>
    <handler type="java:nz.co.lindesay.common.webobjects.LEWOWebServicesWSDLLocationHandler"/>
  </requestFlow>
</transport>
```

You then need to put the following class into your project.

```
package nz.co.lindesay.common.webobjects;

/*
------------------------------------------------------------
LICENSE
------------------------------------------------------------

Copyright (c) 2006, Andrew Lindesay
All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the
following conditions are met:

* Redistributions of source code must retain the above
  copyright notice, this list of conditions and the
  following disclaimer.

* Redistributions in binary form must reproduce the above
  copyright notice, this list of conditions and the
  following disclaimer in the documentation and/or other
  materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
```

```
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

-----------------------------------------------------------
*/

import com.webobjects.foundation.*;
import com.webobjects.appserver.*;

import org.apache.axis.*;

/**
 * <P>This class is an AXIS request handler that is designed to
 * manipulate the outbound WSDL that is generated for a
 * request.  The manipulation undertaken here is to modify the
 * location of the actual service.  The reason for doing this is
 * that if you have a web service deployed into a servlet
 * container as a WO application, the session'ed URLs can look
 * a bit like this.</P>
 *
 * <P><TT>http://foo.co.nz/FOO/WebObjects/FOO.woa/ws/FooService;jsessionid=abc.i1?wsdl</TT></P>
 *
 * <P>However, once the URL is returned as a location for a service
 * from AXIS, the session part at the end is removed and if you are
 * using a multi-container deployment, the subsequent requests will
 * not arrive at the correct servlet container.</P>
 *
 * <P>By installing this AXIS handler, this problem is resolved.
 * To install it, locate the "<TT>server.wsdd</TT>" file in your
 * WO project and add the following line to the "<I>requestFlow</I>"
 * handler list as the last item in the list.  This should be the
 * request flow in the HTTP transport section.</P>
 *
 * <P><TT><handler type="java:nz.co.lindesay.common.webobjects.LEWOWebServicesWSDLLocationHandler"/></TT></P>
 *
 * <P>This should also work with the "<I>wotaskd</I>" deployment
 * style as well.</P>
 */

public class LEWOWebServicesWSDLLocationHandler extends org.apache.axis.handlers.BasicHandler
{

// ------------------------------------------------------------

        public LEWOWebServicesWSDLLocationHandler() { super(); }

// ------------------------------------------------------------

/**
 * <P>This is the AXIS hook where we can manipulate the outbound
 * WSDL.</P>
 */

        public void generateWSDL(MessageContext msgContext) throws AxisFault
        {
                WOContext context = WOWebServiceUtilities.currentWOContext();
                WORequest request = context.request();
                String host = LEWOHelper.getExternalHostnameFromInboundRequest(request);
                int port = LEWOHelper.getExternalPortFromInboundRequest(request);
                int appi = request.applicationNumber();

                if((null==host)||(0==host.length()))
                        throw new IllegalStateException("the host cannot be determined from the inbound
```

```
request.");

                if(context.hasSession())
                        context.session();

                StringBuffer sb = new StringBuffer();

                sb.append("http://");
                sb.append(host);

                if(-1!=port)
                {
                        sb.append(":");
                        sb.append(Integer.toString(port));
                }

                String path = context.urlWithRequestHandlerKey(
                        request.requestHandlerKey(),
                        request.requestHandlerPath(),
                        null);

// Unless the context has a session, we can't get it to come back
// to a specific instance, but in the case of a 'wotaskd' deploy,
// there is no way to specify the session for a WSDL request, so
// we'll do a nasty hack to make this work.

                if((-1!=appi)&&(!context.hasSession()))
                {
                        String webServiceRequestHandlerKey = WOApplication.application().
webServiceRequestHandlerKey();
                        String startPath = request.adaptorPrefix()+"/"+request.applicationName()+".woa"; // ''
/cgi-bin/WebObjects/Foo.woa''

                        if(!path.startsWith(startPath))
                                throw new IllegalStateException("The path '"+path+"' should have started with
'"+startPath+"'");

                        if(path.startsWith(startPath+"/"+webServiceRequestHandlerKey))
                        {
                                StringBuffer pathSB = new StringBuffer();

                                pathSB.append(startPath);
                                pathSB.append("/");
                                pathSB.append(Integer.toString(appi));
                                pathSB.append(path.substring(startPath.length()));

                                path = pathSB.toString();
                        }
                }

                sb.append(path);

                msgContext.setProperty(MessageContext.WSDLGEN_SERV_LOC_URL,sb.toString());
        }

// ------------------------------------------------------------

/**
 * <P>Basically there is nothing to do here because we are not
 * interested in messing with the request/response cycle - rather
 * we want only to manipulate the WSDL.</P>
 */

        public void invoke(MessageContext msgContext) throws AxisFault
        {
        }

// ------------------------------------------------------------

}
```

Although this is slightly crude, it should produce the correct result in the WSDL.