Learn some tips and techniques for troubleshooting the IBM® WebSphere® Application Server V6.1 Feature Pack for Web Services, including common error conditions and suggested methods for correcting them.

The WebSphere Application Server V6.1 Feature Pack for Web Services (hereafter called the WSFP) introduced support for the Java™ API for XML Web Services (JAX-WS) programming model and the Java™ Architecture for XML Binding (JAXB) data binding model. JAX-WS provides a client programming model that includes an API for invoking Web service operations asynchronously, and enables you to use annotations to specify application metadata that would normally appear in deployment descriptor files.

This article describes some basic techniques that can be used to diagnose various problems that might occur while installing, deploying, and executing JAX-WS Web services applications. We’ll look at how to trace application initialization, how to trace SOAP messages, and how to interpret common error conditions.

To get the most from this article, you should have a basic understanding of the WSFP and the JAX-WS programming model, as well as how to use the WebSphere Application Server (hereafter called Application Server) administrative console.

Capturing trace information for JAX-WS applications

Application Server provides a facility for capturing detailed information about the operation and progress of the server and applications. The Application Server Information Center includes
several topics about this tracing facility, including the topic Tracing Web services, which discusses how to enable tracing for the JAX-RPC Web services engine.

Capturing trace information for the JAX-WS Web services engine is similar to the JAX-RPC engine, except that you can use several different trace strings:

**Table 1. JAX-WS trace strings**

<table>
<thead>
<tr>
<th>This trace string:</th>
<th>Enables trace for this component:</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.ws.websvcs.*=all</td>
<td>Integration layer (the set of classes that provide integration of the Axis2/ JAX-WS run-time in the Application Server environment)</td>
</tr>
<tr>
<td>com.ibm.ws.policyset.*=all</td>
<td>Policyset component</td>
</tr>
<tr>
<td>com.ibm.ws.wssecurity.*=all</td>
<td>WS-Security component</td>
</tr>
<tr>
<td>com.ibm.ws.wsaddressing.*=all</td>
<td>WS-Addressing component</td>
</tr>
<tr>
<td>com.ibm.ws.wstx.*=all</td>
<td>WS-Transactions component</td>
</tr>
<tr>
<td>org.apache.axis2.jaxws.*=all</td>
<td>JAX-WS run-time</td>
</tr>
<tr>
<td>org.apache.axis2.*=all</td>
<td>Axis2 run-time (including JAX-WS)</td>
</tr>
</tbody>
</table>

To use the Application Server administrative console to set one or more of these trace strings, do the following:

1. Start Application Server, then open the administrative console.
2. From the main window, select **Application Servers => server1 => Diagnostic Trace Service > Change Log Detail Levels** as shown in Figure 1.
3. In the Properties window, you can select from a list of predefined trace strings, or enter your own trace string directly into the entry field, as shown in Figure 1. For example, to enable tracing for the integration layer and the JAX-WS run-time, you would enter...
com.ibm.ws.websvcs.*=all:org.apache.axis2.jaxws.*=all in the entry field, as shown in Figure 2.

**Figure 2. Turning on tracing in the JAX-WS run-time and in the Application Server integration layer**

Tracing application initialization

One common use of the tracing facility is to examine the details of application initialization. By capturing a trace of your application as it starts, you can make sure that it is initialized properly and that your Web service endpoints are, in fact, available at the endpoint addresses that you intended.

Suppose that you set a trace string for your application server as in Figure 2, and that you have installed the JAX-WS sample application. When the application starts, you would see trace messages similar to Listing 1 in the application server's trace.log file.

Note that the messages in bold show the URL patterns associated with the endpoints contained in your Web service application. These are the URL patterns that client applications use to access those endpoints. For example, if a client application invokes an operation on the `EchoService` endpoint (com.ibm.was.wssample.sei.echo.EchoServicePortImpl implementation class), it would use an endpoint such as `http://MyHost:9080/WSSampleSei/EchoService`.

**Listing 1. Example trace messages showing application initialization**

```
... 
[2/19/08 23:25:39:171 CST] 0000001d WASAxis2Exten 3   Extension Processor is being created for module: SampleServicesSei.war of application: WSSampleServicesSei
...
```
Troubleshooting JAX-WS applications with the WebSphere Application Server V6.1 Feature Pack for Web Services


[com.ibm.was.wssample.sei.ping.PingService12PortImpl]: Initialization successful.

WSWS7037I: The /PingService12 URL pattern was configured for the com.ibm.was.wssample.sei.ping.PingService12PortImpl servlet located in the SampleServicesSei.war web module.

[2/19/08 23:25:39:178 CST] 0000001d WASAxis2Extent 3 Servlet [com.ibm.was.wssample.sei.ping.PingService12PortImpl] has the following mappings:

[0]: com.ibm.was.wssample.sei.ping.PingService12PortImpl:/PingService12
[1]: com.ibm.was.wssample.sei.ping.PingService12PortImpl:/PingService12/wsdl/*
[2]: com.ibm.was.wssample.sei.ping.PingService12PortImpl:*.xsd
[3]: com.ibm.was.wssample.sei.ping.PingService12PortImpl:*.wsdl


...
Tracing SOAP messages

You can also use the tracing facility to trace SOAP messages. Using a trace string of `com.ibm.ws.websvcs.trace.*=all` enables you to view the SOAP messages that flow between the client and server.

Listing 2 shows the request and response messages that result from running the JAX-WS sample application that is included with the WSFP.

**Listing 2. Example of SOAP message tracing**

```
  Content-Type: text/xml
  Message contents:
  <?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body><ns2:echoInput xmlns:ns2="http://com/ibm/was/wssample/sei/echo/">
      This is a test of the JAX-WS sample application
    </ns2:echoInput></soapenv:Body>
  </soapenv:Envelope>
```

This is a test of the JAX-WS sample application.

```
WSWS3569I: Inbound HTTP SOAP request:
  <ns2:echoInput xmlns="http://com/ibm/was/wssample/sei/echo/">
    This is a test of the JAX-WS sample application
  </ns2:echoInput>
```

Content-Type: text/xml; charset=UTF-8
Message contents:
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body><ns2:echoStringInput
xmlns:ns2="http://com/ibm/was/wssample/sei/echo/"><echoInput>
This is a test of the JAX-WS sample application</echoInput></ns2:echoStringInput>
</soapenv:Body></soapenv:Envelope>
WSWS3572I: Outbound HTTP SOAP response:
Content-Type: text/xml
Message contents:
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body><ns2:echoStringResponse
xmlns:ns2="http://com/ibm/was/wssample/sei/echo/">
<echoResponse>JAX-WS==>>
This is a test of the JAX-WS sample application</echoResponse></ns2:echoStringResponse>
</soapenv:Body></soapenv:Envelope>
WSWS3570I: Inbound HTTP SOAP response:
Content-Type: text/xml
Message contents:
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body><ns2:echoStringResponse
xmlns:ns2="http://com/ibm/was/wssample/sei/echo/">
<echoResponse>JAX-WS==>>
This is a test of the JAX-WS sample application</echoResponse></ns2:echoStringResponse>
</soapenv:Body></soapenv:Envelope>

Another way to view SOAP request and response messages is to use the TCPMON utility that is shipped with Application Server. For information on how to use TCPMON, see Tracing SOAP messages with TCPMON in the WebSphere Application Server Information Center.

**Fixing common error conditions**

This section describes some common errors that WSFP users may encounter, and steps you can take to correct them.

**Annotation validation errors within a WAR module**

**Problem:** Annotation validation errors occur during the loading of a WAR module containing JAX-WS Web service endpoints.

**Symptoms:** A WAR module does not start correctly. You might see errors like the following in the SystemOut.log file:

```
Implementation class: com.ibm.was.wssample.sei.echo.EchoService12PortImpl;
missing method name: echoOperation12; endpointInterface:
com.ibm.was.wssample.sei.echo.EchoService12PortType
at org.apache.axis2.jaxws.ExceptionFactory.createWebServiceException(prototype: java:178)
```
Solution: The error message in the log should indicate the area that has failed validation. Consult the logs and make any needed changes to the annotated classes, WSDL, or XSD.

A module contains annotated JAX-WS endpoint classes with clashing @WebService.serviceName values

Problem: Two service implementation bean classes in a single module have the same @WebService.serviceName value, but different @WebService.portName or @WebService.targetNamespace values. The application starts, but only one of the two endpoints can be reached due to the way in which an endpoint’s URL pattern is determined.

Symptoms: Only one of the two endpoints is available via http. The server's SystemOut.log will contain an entry like the following:

Note: This error occurs because the default URL pattern for an implementation class is http://<host>:<port>/<module_context_root>/<@WebService.serviceName>. Since both implementation classes have the same @WebService.serviceName, the Web container reports a mapping clash and the default URL pattern cannot be established for the second endpoint.

Solution: Define unique URL patterns for the different endpoints in the module’s web.xml file. You can find more information about how to do this in the Information Center topic Customizing URL patterns in the web.xml file for JAX-WS applications.

Application fails to start because of problems finding WSDL or schema files

Problem: An application fails to start due to a FileNotFoundException. This happens when a WSDL or schema (.xsd) file can't be found.

Symptoms: An error message like the following will appear in the server's SystemOut.log file:
Troubleshooting JAX-WS applications with the WebSphere Application Server V6.1 Feature Pack for Web Services

0000001c WASAxis2CompoE WSW7007E: The SampleServicesSei.war application module cannot be loaded correctly because of the following error: java.lang.Exception: Configuration data could not be created for the Web service related class
com/ibm/was/wssample/sei/echo/EchoService12PortImpl:
com.ibm.ws.websvcs.exception.DeploymentException:
java.io.FileNotFoundException:
C:\was\ws\profiles\AppSrv01\installedApps\IBM-D81F283B1E1Node01Cell\WSSampleServicesSei.ear\SampleServicesSei.war\WEB-INF\wsdl\Ech.wsdl
(The system cannot find the file specified.)

Solution: Ensure that the WSDL document referenced by the Web service implementation's @WebService.wsdlLocation value is in the specified location, and that any imported WSDL and schema (.xsd) files are in their specified locations.

JAX-WS service clients do not appear in the administrative console Service Clients list

Problem: JAX-WS service clients in a deployed WAR module do not appear in the administrative console Service Clients list.

Symptoms: The client functions properly, but doesn't appear in the administrative console list. This prevents users from attaching policy sets to the client using the administrative console. A com.ibm.ws.websvcs.*=all trace would yield a warning message like the following:

WSModuleDescr W WSW7062E: The c:/wsdl/WEB-INF/wsdl/Echo12.wsdl Web Services Description Language (WSDL) file referenced by the @WebServiceClient annotation in the com/ibm/was/wssample/sei/echo/EchoService12 class could not be loaded

If the client does not show up in the console listing, and there is no warning message in the server's trace log like the message above, ensure that the JAX-WS client has an @WebServiceClient annotation. If the client was properly recognized by the run-time, the following trace message will be logged: @WebServiceClient annotation processed: EchoService.

Solution: Although this won't prevent the client from functioning properly, it limits the ability of users to configure the client with policy sets using the administrative console. To ensure that a JAX-WS client appears in the administrative console, ensure that it is annotated with @WebServiceClient and that the wsdlLocation attribute value specifies a valid WSDL document.

Connection refused error when invoking a Web service

Problem: An application containing a JAX-WS Web service was successfully installed, but attempts to invoke the Web service fail.

Symptoms: The client receives the following error and the Web service invocation fails:
javax.xml.ws.WebServiceException: java.net.ConnectException: 
Connection refused: no further information
at org.apache.axis2.jaxws.ExceptionFactory.createWebServiceException
(ExceptionFactory.java:180)
at org.apache.axis2.jaxws.ExceptionFactory.makeWebServiceException
(ExceptionFactory.java:79)
at org.apache.axis2.jaxws.ExceptionFactory.makeWebServiceException
(ExceptionFactory.java:134)
at org.apache.axis2.jaxws.core.controller.AxisInvocationController.execute
(AxisInvocationController.java:595)

Solution: First, make sure that the client is using the correct endpoint to invoke the Web service (correct host, port, context root, and URL pattern). It's also possible that the application installed successfully, but did not start successfully. Errors that occurred during the start of the application would be present in the SystemOut.log or SystemErr.log, as well as the trace.log, if trace was enabled. A Web services trace (com.ibm.ws.websvcs.*=all) of the server can help you determine whether the application is starting correctly and which URL patterns were used with the various endpoints. Here is an example:

Added mapping [/PingService12] for servlet
[com.ibm.was.wssample.sei.ping.PingService12PortImpl]

This indicates that the WASAxis2ExtensionProcessor class has created a URL pattern /PingService12, which is associated with the Web service implementation class com.ibm.was.wssample.sei.ping.PingService12PortImpl.

The asynchronous response servlet does not start properly

Problem: The async response servlet does not start properly.

Symptoms: You'll see error messages in the SystemOut.log file similar to these:

org.apache.axis2.AxisFault:
WSWS7115E: The required AsyncResponseServlet is not available.
getEPRsForService(AsyncResponseServletListener.java:114)
at org.apache.axis2.description.OutInAxisOperationClient.execute
(OutInAxisOperation.java:243)

Solution: This type of error normally indicates that the async response servlet application (ibmasyncrsp.ear) was not properly installed, which implies that the Application Server's profile was not created correctly.
1. Make sure that the async response servlet application is installed and starts up properly. You should see messages in the SystemOut.log similar to these when the Application Server starts up:

```
[8/15/07 13:37:16:437 CDT] 00000013 ApplicationMg A
WSVR0200I: Starting application: ibmasyncrsp
[8/15/07 13:37:16:437 CDT] 00000013 ApplicationMg A
WSVR0203I: Application: ibmasyncrsp Application build level: WSPF.WSERV1 [x0724.11]
SRVE0169I: Loading Web Module: WebSphere ASYNC Response Servlet Application.
SRVE0242I: [ibmasyncrsp] [/IBM_WS_SYS_RESPONSESERVLET] [rspservlet]: Initialization successful.
[8/15/07 13:37:17:703 CDT] 00000013 ApplicationMg A
WSVR0221I: Application started: ibmasyncrsp
```

2. Make sure that your application server profile was created correctly.

### Inconsistent SSL configuration

**Problem:** A client tries to access a Web service that has been configured with SSL, but the client does not have SSL configured.

**Symptoms:** You will see error messages similar to the following:

```
exception01 java.security.PrivilegedActionException:
org.apache.axis2.AxisFault: WSWS7130E: No Secure Sockets Layer (SSL) configuration is available for the
at java.security.AccessController.doPrivileged
(AccessController.java:246)
at com.ibm.ws.security.util.AccessController.doPrivileged
(AccessController.java:125)
setupEffectiveSSLConfiguration(SetupSSLConfiguration.java:69)
setupHTTPConnection(SOAPOverHTTPSender.java:1483)
is302or401set(SOAPOverHTTPSender.java:952)
send(SOAPOverHTTPSender.java:377)
```

**Solution:** Make sure that SSL is configured on the client.

### Web service invocation times out

**Problem:** The client encounters a timeout when waiting for a response message to arrive from the server.

**Symptoms:** You will see error messages similar to the following:
Caused by: java.net.SocketTimeoutException: Async operation timed out
    processSyncReadRequest(AioTCPReadRequestContextImpl.java:157)
    at com.ibm.ws.tcp.channel.impl.TCPReadRequestContextImpl.
    read(TCPReadRequestContextImpl.java:109)
    HttpOutboundServiceContextImpl.parseResponseMessageSync
    (HttpOutboundServiceContextImpl.java:1627)
    HttpOutboundServiceContextImpl.readSyncResponse
    (HttpOutboundServiceContextImpl.java:702)
    HttpOutboundServiceContextImpl.startResponseReadSync
    (HttpOutboundServiceContextImpl.java:1745)
    HttpOutboundServiceContextImpl.finishRequestMessage
    (HttpOutboundServiceContextImpl.java:1164)
    sendSOAPRequest(SOAPOverHTTPSender.java:492)

Solution: Try to determine why it's taking so long for the response to arrive at the client. If it's
due to a problem on the server, you should be able to determine this by looking at the serverâs
SystemOut.log or trace.log. If the excessive time is due to the request simply taking a long time
to complete on the server, you need to increase the readTimeout value in the HTTP transport
PolicySet.

Summary

In this article, you learned some basic problem determination techniques you can use with the
Feature Pack for Web Services. In addition, you learned about some typical error conditions and
some possible solutions to those errors. This article can help you quickly recognize and diagnose
problems that you encounter while developing, deploying, and executing your JAX-WS Web
service applications using the Feature Pack for Web Services.
Related topic

• WebSphere Application Server 6.1.0 documentation

© Copyright IBM Corporation 2008
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)