WebSphere® Application Server Network Deployment for IBM i, Version 7.0

IBM

Troubleshooting WebSphere applications

Note

Before using this information, be sure to read the general information under “Notices” on page 153.

Compilation date: September 16, 2008

Contents

How to send your comments v

Changes to serve you more quickly vii

Chapter 1. SIP applications 1
Tracing a Session Initiation Protocol proxy server 1
Tracing a SIP container 1

Chapter 2. Web services 3
Troubleshooting Web services 3
Web services command-line tools troubleshooting tips 3
Web services compiled bindings troubleshooting tips 7
Web services client runtime troubleshooting tips 7
Web services serialization and deserialization troubleshooting tips 10
Web services authentication, authorization and secure transport troubleshooting tips. 12
Application client sending SOAP request receives errors 13
Universal Discovery, Description, and Integration, Web service, and SOAP component troubleshooting tips 14
Tracing Web services 15
Tracing SOAP messages with tcpmon 17
Frequently asked questions about Web services 18
Web services security troubleshooting tips 20
Trace and logging for WSIF. 31
UDDI registry troubleshooting 32
UDDI registry common errors 32
UDDI registry problem reporting 33

Chapter 3. Service integration 35
Resolving indoubt transactions 35
Restoring a data store and recovering its messaging engine. 36
Messaging engine failover between Version 6.x and Version 7.0 38
Problem solving for messaging engine data stores 38
Diagnosing problems with data store exclusive access locks 38
Diagnosing problems with your data store configuration 40
Avoiding failover problems when you use DB2 v8.2 with HADR as your data store 40
Problem solving for messaging engine file stores 40
Diagnosing problems with accessing file store files 41
Reducing file store file sizes 41
Listing messages on a message point 42
Deleting messages on a message point 42
Troubleshooting service integration message problems 43
Understanding why best effort messages are being discarded 43
Investigating why a queue is full 43
Investigating why a topic space is full 45
Investigating why point-to-point messages are not arriving 47
Investigating why point-to-point messages are not being consumed 52
Investigating why publish/subscribe messages are not arriving at a subscription 59
WS-Notification troubleshooting tips. 64
WS-Notification: Known restrictions 71

Chapter 4. Data access resources 73
Using a single instance of a resource adapter 73

Chapter 5. Messaging resources . . . 75
Troubleshooting WebSphere messaging . . . 75
Messaging troubleshooting tips . . . 76
Troubleshooting message-driven beans . . . 81
Troubleshooting performance monitoring statistics . . . 82

Chapter 6. Mail, URLs, and other J2EE resources . . . 83
Debugging a mail session . . . 83

Chapter 7. Security . . . 87
Troubleshooting security configurations . . . 87
Security components troubleshooting tips. . . . 87
Security configuration and enablement errors . . . 98
Security enablement followed by errors . . . 101
Access problems after enabling security. . . . 106
Secure Sockets Layer errors . . . 111
Single sign-on configuration troubleshooting tips. . . . 114
Enterprise Identity Mapping troubleshooting tips . . . 117
Security authorization provider troubleshooting tips. . . . 119
Password decoding troubleshooting tips. . . . 123
SPNEGO trust association interceptor (TAI) troubleshooting tips (deprecated) . . . 123

Chapter 8. Naming and directory . . . 129
Troubleshooting namespace problems . . . 129
Naming service troubleshooting tips . . . 129
Application access problems . . . 130
Viewing a namespace dump . . . 134
dumpNameSpace tool . . . 136
Viewing `java:`, `local:` and `server` namespace dumps . . . 139
Namespace dump utility for `java:`, `local:` and `server` namespaces . . . 141

Chapter 9. Transactions . . . 143
Troubleshooting transactions . . . 143
Transaction troubleshooting tips. . . . 143

Chapter 10. Learn about WebSphere programming extensions . . . 145
ActivitySessions . . . 145
Troubleshooting ActivitySessions . . . 145
Dynamic cache . . . 145
Troubleshooting the dynamic cache service . . . 145

Appendix. Directory conventions . . . 151

Notices . . . 153

Trademarks and service marks . . . 155

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-5250.

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Changes to serve you more quickly

Print sections directly from the information center navigation

PDF books are provided as a convenience format for easy printing, reading, and offline use. The information center is the official delivery format for IBM WebSphere Application Server documentation. If you use the PDF books primarily for convenient printing, it is now easier to print various parts of the information center as needed, quickly and directly from the information center navigation tree.

To print a section of the information center navigation:

1. Hover your cursor over an entry in the information center navigation until the **Open Quick Menu** icon is displayed beside the entry.
2. Right-click the icon to display a menu for printing or searching your selected section of the navigation tree.
3. If you select **Print this topic and subtopics** from the menu, the selected section is launched in a separate browser window as one HTML file. The HTML file includes each of the topics in the section, with a table of contents at the top.
4. Print the HTML file.

For performance reasons, the number of topics you can print at one time is limited. You are notified if your selection contains too many topics. If the current limit is too restrictive, use the feedback link to suggest a preferable limit. The feedback link is available at the end of most information center pages.

Under construction!

The Information Development Team for IBM WebSphere Application Server is changing its PDF book delivery strategy to respond better to user needs. The intention is to deliver the content to you in PDF format more frequently. During a temporary transition phase, you might experience broken links. During the transition phase, expect the following link behavior:

- Links to Web addresses beginning with http:// work
- Links that refer to specific page numbers within the same PDF book work
- The remaining links will *not* work. You receive an error message when you click them

Thanks for your patience, in the short term, to facilitate the transition to more frequent PDF book updates.

Chapter 1. SIP applications

Tracing a Session Initiation Protocol proxy server

You can trace a Session Initiation Protocol (SIP) proxy server, starting either immediately or after the next server startup.

About this task

To trace a SIP proxy server, complete the following steps:

1. Start the product, and open the administrative console.
2. In the administrative console, click **Troubleshooting** → **Logs and trace**.
3. Select the name of the server for the SIP proxy server.
4. From the General Properties section, click **Diagnostic Trace Service**.
5. Select one of the following options:

Option	Description
Configuration	To start tracing after the next server startup
Runtime	To start tracing immediately

6. Replace the content of the trace specification with the following code: `*=info:com.ibm.ws.sip.*=all:com.ibm.ws.proxy.*=all`.
7. Make sure that the **Enable trace with following specification** check box is checked.
8. Click **Apply** → **Save**.

What to do next

When the changes take effect, SIP proxy server tracing messages display in *WASProductDir*/logs/*serverName*/trace.log on the SIP proxy server node, where *WASProductDir* is the fully-qualified path name of the directory where the product is installed and *serverName* is the name of the specific instance of the application server that is running the SIP proxy server to be traced.

Tracing a SIP container

You can trace a Session Initiation Protocol (SIP) container, starting either immediately or after the next server startup. This tracing writes a record of SIP events to a log file.

About this task

Follow these steps to start tracing a SIP container:

1. Open the administrative console. For more information about the console, read the Using the administrative console chapter of the *Administering applications and their environment* PDF book.
2. In the administrative console, click **Troubleshooting** → **Logs and trace**.
3. Select the name of the server for the SIP container.
4. From the General Properties section, click **Diagnostic Trace Service**.
5. Under the Additional Properties section, click **Change Log Detail Levels**
6. Select one of the following options:

Option	Description
Configuration	To start tracing after the next server startup

Option	Description
Runtime	To start tracing immediately

7. Replace the content of the trace specification with the following code: `com.ibm.ws.sip.*=all=enabled`.

   Note: If you want monitor only specific pieces of SIP containers, expand the **com.ibm.ws.sip** section and select the individual items you wish to trace.
8. Make sure that the **Enable trace with following specification** check box is checked.
9. Click **Apply** → **Save**.

What to do next

When the changes take effect (refer to step 6 above), SIP-level tracing messages appear in *WASProductDir*/logs/*serverName*/trace.log, where *WASProductDir* is the fully qualified path name of the directory in which the product is installed and *serverName* is the name of the specific instance of the application server that is running the SIP container to be traced. These messages include application load events as well as SIP request and response parsing and SIP servlet invocation.

Chapter 2. Web services

Troubleshooting Web services

Learn about ways that you can troubleshoot Web services applications.

Before you begin

This topic provides information for you to troubleshoot during different steps of the development, assembly, deployment, and security processes of a Web service.

About this task

Select the Web services topic area that you want to troubleshoot:

- Command-line tools

  This topic provides information on troubleshooting the WSDL2Java command-line tool and the Java2WSDL command-line tool.
- Java™ compiler errors

  This topic discusses troubleshooting compiled bindings of Web services.
- Serialization or deserialization errors

  This topic presents problems you might encounter performing serialization and deserialization in Web services.
- Authentication challenges and authorization failures with Web services security

  This topic discusses troubleshooting authentication and authorization when you are securing Web services.

Web services command-line tools troubleshooting tips

This topic discusses troubleshooting the **WSDL2Java** and **Java2WSDL** command-line tools that are used when you develop Java API for XML-based RPC (JAX-RPC) Web services.

Each section in this topic is a problem that you might experience while using the WSDL2Java or Java2WSDL tool. A solution is provided to help you troubleshoot the problem.

The .Net client does not reflect a Web service method with Vector-type parameters

The following exception displays when running the .NET client for a Web service:

```
System.InvalidOperationException: Method AnnuityInteropService.wsListAnnuityByHolder cannot be reflected.
System.InvalidOperationException: There was an error reflecting wsListAnnuityByHolderResult.
System.InvalidOperationException: The Form property might not be unqualified when an explicit namespace
property is available.
```

The problem is exposed when the server-side method returns a Vector and the style of the Web Services Description Language (WSDL) file is document/literal and the Form is unqualified. The unqualified Form is always generated for the document/literal style because elementFormDefault=″unqualified″ by default.

A problem exists in the .NET framework that can generate a Web service proxy class that is not valid when your Web service methods contain certain arrays as parameters. The framework adds an XmlArrayAttribute attribute to the parameter, and the attribute constructor contains an ″Form=Unqualified″ and a namespace definition. While the attribute is valid, it is not valid to combine these two aspects of the attribute, causing the `System.InvalidOperationException` exception.

To avoid this problem, do not use Vector parameters in a Web service method that is deployed into WebSphere® Application Server. Vector parameters in a Web service method do not work with a .Net client or a Java collection type.

Error with the Endpoint Enabler tool on a Hyper Threading-enabled machine

When you run the Endpoint Enabler tool on a Hyper Threading-enabled machine, an error is reported in the Tasks view against the Enterprise Application project. The Enterprise Application project cannot be deleted until you restart the workbench.

This problem can occur if a race condition exists between the automatic build and the Enterprise JavaBeans™ (EJB) project validator.

To resolve the problem, turn off the workbench automatic build before running the Endpoint Enabler tool, and turn it back on afterwards.

Multiprotocol port component restrictions with JSR109 Version 1.0 and 1.1

Java Specification Requests (JSR) 109 specification validation errors occur when deploying an enterprise archive (EAR) file that contains a WSDL file with `http`, `jms` and `ejb` bindings generated by the **Java2WSDL** command-line tool.

The JSR 109 specification requires each port component defined in the `webservices.xml` deployment descriptor file to refer to unique <servlet-class> elements in `web.xml` file for a JavaBeans implementation, or a unique <session> element in `ejb-jar.xml` file for an EJB implementation. The servlet and session EJB are located in the `webservices.xml` file represented by <servlet-link> or <ejb-link> element.

The **WSDL2Java** command-line tool maps the ports found in a WSDL file to port components that are in the generated `webservices.xml` file. If a single Web service has multiple bindings, in addition to a port for each of these bindings, the `webservices.xml` file contains multiple port components that should all point to the same EJB module(<session>) or JavaBeans (<servlet-class>) implementation. Because of the JSR 109 restrictions, the `webservices.xml` file is not valid and errors can occur during the deployment process.

The following example displays the error:

```
Error in <module> : CHKW6030E: Implementation class <class> referred to by port
components<port1> and <port2>. (JSR109 1.0: 7.1.2).
```

Here is the error with sample data:

```
Error in WebSvcsInSession20EJB.jar : CHKW6030E: Implementation class WSMultiProtocol
referred to by port components WSMultiProtocolJMS and WSMultiProtocolEJB.(JSR109 1.0: 7.1.2).
```

You can work around the restriction by creating multiple <session> EJB definitions within the `ejb-jar.xml` file that all point to the same implementation class, home interface and remote interface. You can still use the same classes, but the `ejb-jar.xml` file <session> definitions that reference the classes and the interfaces must be duplicated.

The following is an example of the `webservices.xml` file. Look for the classes and interfaces:

```
<webservices>
  <webservice-description>
    <webservice-description-name>WSMultiProtocolService</webservice-description-name>
    <wsdl-file>META-INF/wsdl/WSMultiProtocol.wsdl</wsdl-file>
    <jaxrpc-mapping file>META-INF/WSMultiProtocol_mapping.xml</jaxrpc-mapping file>
    <port-component>
      <port-component-name>WSMultiProtocolEjb</port-component-name>
      <wsdl-port>
        <namespaceURI>http://ejb.pli.tc.wssvt.ibm.com</namespaceURI>
        <localpart>WSMultiProtocolEjb</localpart>
      </wsdl-port>
      <service-endpoint-interface>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol
   </service-endpoint-interface>
      <service-impl-bean>
        <ejb-link>WSMultiProtocol</ejb-link>
      </service-impl-bean>
    </port-component>
    <port-component>
  <port-component-name>WSMultiProtocolJMS</port-component-name>
      <wsdl-port>
        <namespaceURI>http://ejb.pli.tc.wssvt.ibm.com</namespaceURI>
        <localpart>WSMultiProtocolJMS</localpart>
      </wsdlport>
      <service-endpoint-interface>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol
   </service-endpoint-interface>
      <service-impl-bean>
        <ejb-link>WSMultiProtocol_2</ejb-link>
      </service-impl_bean>
  </port-component>
        <port-component>
         <port-component-name>WSMultiProtocolJMS</port-component-name>
      <wsdl-port>
        <namespaceURI>http://ejb.pli.tc.wssvt.ibm.com</namespaceURI>
        <localpart>WSMultiProtocolJMS</localpart>
      </wsdlport>
      <service-endpoint-interface>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol
   </service-endpoint-interface>
      <service-impl-bean>
        <ejb-link>WSMultiProtocol_3</ejb-link>
      </service-impl_bean>
  </port-component>
 </webservice-description>
</webservices>
```

The following is an example of the `ejb-jar.xml` file. Look for the classes and interfaces, and how they are duplicated:

```
<ejb-jar-id="ejb-jar_ID">
 <display-name>WebSvcsInsSession20EJB</display-name>
 <enterprise-beans>
  <session-id="WSMultiProtocol">
   <ejb-name>WSMultiProtocol</ejb-name>
   <home>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolHome</home>
   <remote>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol</remote>
   <ejb-class>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolWebSvcsBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
   <ejb-ref-id="EjbRef_1082407586720">
    <description></description>
    <ejb-ref-name>ejb/BeneficiarySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.BeneficiarySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.BeneficiarySession</remote>
    <ejb-link>PolicySession20EJB.jar#BeneficiarySession</ejb-link>
   </ejb-ref>
   <ejb-ref-id="EjbRef_1082407586790">
    <description></description>
    <ejb-ref-name>ejb/PolicySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.PolicySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.PolicySession</remote>
    <ejb-link>PolicySession20EJB.jar#PolicySession</ejb-link>
   </ejb-ref>


  <session-id="WSMultiProtocol_2">
   <ejb-name>WSMultiProtocol_2</ejb-name>
   <home>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolHome</home>
   <remote>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol</remote>
   <ejb-class>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolWebSvcsBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
   <ejb-ref-id="EjbRef_1082407586720_2">
    <description></description>
    <ejb-ref-name>ejb/BeneficiarySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.BeneficiarySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.BeneficiarySession</remote>
    <ejb-link>PolicySession20EJB.jar#BeneficiarySession</ejb-link>
   </ejb-ref>
   <ejb-ref-id="EjbRef_1082407586790_2">
    <description></description>
    <ejb-ref-name>ejb/PolicySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.PolicySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.PolicySession</remote>
    <ejb-link>PolicySession20EJB.jar#PolicySession</ejb-link>
   </ejb-ref>


  <session-id="WSMultiProtocol_3">
   <ejb-name>WSMultiProtocol_3</ejb-name>
   <home>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolHome</home>
   <remote>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol</remote>
   <ejb-class>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolWebSvcsBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
   <ejb-ref-id="EjbRef_1082407586790_3">
    <description></description>
    <ejb-ref-name>ejb/BeneficiarySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.BeneficiarySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.BeneficiarySession</remote>
    <ejb-link>PolicySession20EJB.jar#BeneficiarySession</ejb-link>
   </ejb-ref>
   <ejb-ref-id="EjbRef_1082407586790_3">
    <description></description>
    <ejb-ref-name>ejb/PolicySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.PolicySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.PolicySession</remote>
    <ejb-link>PolicySession20EJB.jar#PolicySession</ejb-link>
   </ejb-ref>
  </session>
```

Avoiding application errors after uninstalling an interim fix, a fix pack, or a refresh pack

If an application uses functions that are provided by a particular fix and you remove the fix, the application displays an error message. If you remove a fix, make sure that you retest your applications to check for errors. Redeploy any applications that display an error message because of the missing fix.

For example, suppose you install a fix pack on WebSphere Application Server and you create the stock quote Web service, StockQuote. The **WSDL2Java** command-line tool is used in a deployer role and generates a ServiceLocator class that extends the AgnosticService class.

If you uninstall the fix pack, the application is using a new Web services class (AgnosticService) that the version of WebSphere Application Server does not support. The application creates the following error:

```
java.lang.NoClassDefFoundError:
 Error while defining class:
 com.ibm.ws.wsfvt.test.stockquote.StockQuoteServiceLocator
 This error indicates that the class:
 com.ibm.webservices.multiprotocol.AgnosticService
 could not be located  while defining the class:
 com.ibm.ws.wsfvt.test.stockquote.StockQuoteServiceLocator
```

You need to redeploy the application on the WebSphere Application Server to emit code that does not use the WebSphere Application Server version that is not supported by the Web services class that you use.

Using a proxy server to access the Internet while running the WSDL2Java command causes your connection to time out

If you use an environment that requires a proxy server to access the Internet during the run of the **WSDL2Java** command, the **WSDL2Java** command might not find the Internet information because the proxy server has the potential to time out. For example, if the input WSDL file is located on the Internet instead of a local drive, and you need to retrieve it from the Internet, the **WSDL2Java** command fails to find the file because the proxy server times out.

You can work around this problem by editing the **WSDL2Java** command that is located in the `app_server_root/bin` directory. Set your proxy host and port value in one of following ways:

- Edit the `profile_root/bin/WSDL2Java` file and add the following line to the beginning of the file:

```
export PROXY_INFO="-Dproxy.httpHost=yourProxyHost -Dproxy.httpPort=yourProxyPort"
```

  If you use this option, the **WSDL2Java** command, located in the `profile_root/bin` directory, must be invoked.

- Set the `PROXY_INFO` environment variable in the Qshell session prior to invoking the **WSDL2Java** command as follows:

```
$ export PROXY_INFO ="-Dproxy.httpHost=yourProxyHost -Dproxy.httpPort=yourProxyPort"
```

If you use this option, you need to run the command every time a new QSHELL session is started. If you do not want to start the QSHELL each time, you can add the command to the profile.

Web services compiled bindings troubleshooting tips

This topic discusses troubleshooting compiled bindings of Web services that are developed and implemented based on Java programming models.

Each section in this topic is a problem that you might experience with compiled bindings for Web services. A solution is provided to help you troubleshoot the problem.

Context root not recognized when mapping the default XML namespace to a Java package

When you map the default XML namespace to a Java package the context root is not recognized. If two namespaces are the same up to the first slash, they map to the same Java package. For example, the XML namespaces `http://www.ibm.com/foo` and `http://www.ibm.com/bar` both map to the `www.ibm.com` Java package . Use the -NStoPkg option of the **Java2WSDL** command to specify the package for the fully qualified namespace.

Java code to Web Service Description Language (WSDL) mapping cannot be reversed to the original Java code

If you find that a WSDL file that you created with the **Java2WSDL** command-line tool cannot be compiled when regenerated into Java code using the **WSDL2Java** command-line tool, it is because the Java API for XML-based remote procedure call (JAX-RPC) mapping from Java code to WSDL is not reversible back to the original Java code.

To troubleshoot this problem, try specifying the -introspect option to the **WSDL2Java** command. The -instrospect option indicates to the **WSDL2Java** command to look into existing Java classes and gather information useful in generating artifacts that match the original Java code.

The session bean fails to instantiate when the Web service is accessed

If you are trying to access a Web service and you get the following error, `WSWS3422E: Error: Can not instantiate bean_name`, the session bean might be trying to be accessed as a servlet-type Web service.

If this error message displays during the initial testing of a Web service, you need to verify with the Web service developer that the correct type of Web service was generated. For example, if a session bean is exposed as a Web service, an enterprise bean-type Web service is created. A session bean that is accessed as a servlet-type Web service can cause this exception.

Web services client runtime troubleshooting tips

Use these tips to troubleshoot Web services clients.

Each section in this topic is a problem that you might experience during the run time of a Web services client. A solution is provided to help you troubleshoot the problem.

Use the ASYNC_TIMEOUT_MILLISECONDS property to avoid receiving a time out exception for JAX-WS synchronous clients

If your Java API for XML-based Web Services (JAX-WS) synchronous clients receive the Web services exception, `org.apache.axis2.AxisFault: Time out while waiting for the server to send the response`, set the asynchronous timeout property, `com.ibm.websphere.webservices.jaxws.Constants.ASYNC_TIMEOUT_MILLISECONDS` , on the client to control the amount of time to wait for the response from the server before a time out error message is generated.

Specify the timeout property in milliseconds to set the amount of time to wait for a reply to an asynchronous request. The following example demonstrates how to set this property:

```
rc.put("com.ibm.websphere.webservices.jaxws.Constants.ASYNC_TIMEOUT_MILLISECONDS", 30000);
```

The connection to the remote host fails

If the following error, `WSWS3713E: Connection to the remote host` *`host_name`* `failed`, displays when you are trying to connect to the remote host, check the following items:

- If the host name that is listed in the error message is the correct host name, you need to verify that the application with the Web service is running and is available.
- If the host name that is listed in the error message is the incorrect host name, you might need to update the WSDL file for the Web service or override the endpoint URL that the host name needs to use. To override the Web service endpoint URL, see the information center topic ″Configuring Web services client bindings.″ You can configure the port information with the information provided in this topic.

Running your Web service client with an `ibm-jaxrpc-client.jar` file in a Solaris environment can cause an exception

If you use the **-jar** option, for example, `java -jar <java_application>.jar`, to specify a Java application in a Solaris environment, a class not found exception can occur. To avoid an exception, use the **-classpath** option instead of the **-jar** option, for example,

```
java -jar <your_client_application>.jar, <main_class_file>
```

.

The problem occurs because the Sun JDK classloading specification is more strict than the IBM® JDK specifications.

You can make one of three changes to avoid an exception:

- Use the **-classpath** option instead of the **-jar** option, for example,

  ```
  java -jar <java_application>.jar, <main_class_file >
  ```

  .
- Use the **-Djava.ext.dirs** option with the **-jar** option, for example,

  ```
  export WAS_HOME=/opt/IBM/WebSphere/AppServer ${WAS_HOME}/java/jre/bin/java
  -Djava.ext.dirs=${WAS_HOME}/runtimes
  -jar <your_client_application>.jar, <your_client_application>.args
  ```

  .
- Modify Class Path in Manifest.MF to include the Java archive (JAR) files that you need, for example,

  ```
  Class Path: /opt/IBM/WebSphere/AppServer/runtimes/ibm-jaxrpc-client.jar
  ```

Resolving DNS causes performance problems when using HTTP to connect to a service endpoint interface that is not based on a private IP address

The DNS service is often not available when you use HTTP to connect to a service endpoint interface that is based on a private IP address. Therefore, performance is degraded during the DNS resolution.

This problem occurs when the outbound HTTP connector in the Web service engine attempts to resolve the host address name and times out.

You can modify the HOSTS file for the targeted IP address to avoid the DNS resolution.

Runtime migration error

If you installed a Web service application that was developed for a WebSphere Application Server version prior to Version 6, you might get the following exception:

```
WSWS3701E: Error: An exception was encountered. Use wsdeploy to deploy your application.
This might correct the problem. The exception is <exception data>.
```

This exception indicates that a problem occurred while running the application that was developed with tools supported by versions prior to Version 6. A solution to the problem is to uninstall the application, run the **wsdeploy** command and redeploy the application.

The wsdeploy command is supported by Java API for XML-based RPC (JAX-RPC) applications. The Java API for XML-Based Web Services (JAX-WS) programming model that is implemented by the application server does not support the wsdeploy command. If your Web services application contains only JAX-WS endpoints, you do not need to run the wsdeploy command, as this command is used to process only JAX-RPC endpoints.

WebServicesFault exception displays during the application server run time for certain Web Services Description Language (WSDL) files

A `WebServicesFault` exception displays during the application server run time for WSDL files that define operations with `document style` and `literal use`, and use the SOAP header to transmit the input data.

If the WSDL files define the operation with `document style` and `literal use`, and this operation maps the input to the SOAP header, the Web services run time fails to find the correct operation for the target service and the `WebServicesFault` exception displays.

To solve the problem, change the WSDL files so that the operation does not have input that uses the SOAP header to transmit the data.

Increase the value of the ConnectionIOTimeOut parameter to avoid receiving an exception when hosting Web services

When hosting Web services on WebSphere Application Server, the following exception displays: `java.net.SocketTimeOutException: Read Timed Out`.

A slow network connection between the client and the Web service causes this problem. In such cases, the HTTP socket might time out before the Web service engine completely reads the SOAP request. In the majority of cases, a sudden increase in overall network activity causes this problem. The problem can also occur when the client is accessing the Web service from a slow network connection and when the SOAP request has a lot of data.

To solve the problem, increase the ConnectionIOTimeOut parameter for the Web container HTTP transport. The default value is 5 seconds. Increase the value to 30 seconds or greater. Type the following property name and value:

- **Name:** ConnectionIOTimeOut
- **Value:** 30

If the Web service is hosted in a clustered environment, set the property on each application server in the cluster. If your application server is listening on more than one port number, set the property on all ports. For more information about setting the value using the administrative console, refer to the HTTP transport custom properties chapter in the *Administering applications and their environment* PDF book.

Increase the value of the syncTimeout parameter to avoid receiving an exception when hosting Web services clients

You can also get the `java.net.SocketTimeOutException: Read Timed Out` error when the syncTimeout parameter that is used by the Web services client is not set correctly. This is important to know because if you set the ConnectionIOTimeout parameter to zero with the expectation that a timeout is preventable as stated in the topic ″HTTP transport custom properties″ only the connection timeout is prevented. The only way to make sure that a request from an HTTP client, which can be a Web services client, does not time out, is to increase the syncTimeout parameter setting.

The syncTimeout parameter is only used by the Web services client. This parameter can be set in the Web services stub that is a timeout for the Web services call.

Executing a Web services client application with session persistence turned on or in a clustered environment might cause a WebServicesFault error

When you run a Web services client application with session persistence turned on or in a cluster environment, an error might display because the Web service client attempts to use a connection that has been closed by the HTTP server. The following is an example of the error:

```
[mm/dd/yy hh:mm:ss:ttt EST] 0000006e SystemErr     R WebServicesFault
 faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.generalException
 faultString: java.io.IOException: Connection close: Read failed.Possible end of
stream encountered.
 faultActor: null
 faultDetail:
```

You can avoid this error by following one of two ways:

- Set the com.ibm.websphere.webservices.http.requestResendEnabled property to `true`, for example, com.ibm.websphere.webservices.http.requestResendEnabled=true. When this property is set to `true`, the Web services client is programmed to re-send the request if the request has failed. Monitor your client runtime if you change the property value, because the request might be sent twice.

  For example, if your client is a banking application, and you set the com.ibm.websphere.webservices.http.requestResendEnabled property to `true`, a transaction might be posted twice to an account. For more detailed information on configuring this property, read the section, Configuring additional HTTP transport properties using the JVM custom property panel in the administrative console in the *Developing and deploying applications* PDF book.

Web services serialization and deserialization troubleshooting tips

This topic discusses problems that you can have when you perform serialization and deserialization in Web services.

Each section in this topic is a problem that you might experience while serializing and deserializing Web services. A solution is provided to help you troubleshoot the problem.

Time zone information in deserialized java.util.Calendar is not as expected

When the client and server are based on Java code and a `java.util.Calendar` instance is received, the time zone in the received `java.util.Calendar` instance might be different from that of the `java.util.Calendar` instance that was sent.

This difference occurs because the `java.util.Calendar` is encoded as an xsd:dateTime for transmission. An xsd:dateTime is required to encode the correct time (base time plus or minus a time zone offset), but is not required to preserve locale information, including the original time zone.

The fact that the time zone for the current locale is not preserved needs to be accounted for when comparing Calendar instances. The `java.util.Calendar` class equals method checks that the time zones are the same when determining equality. Because the time zone in a deserialized Calendar instance might not match the current locale, use the before and after comparison methods to test that two Calendars refer to the same date and time as shown in the following examples:

```
java.util.Calendar c1 = ...// Date and time in time zone 1
java.util.Calendar c2 = ...// Same date and equivalent time, but in time zone 2

// c1 and c2 are not equal because their time zones are different
if (c1.equals (c2)) System.out.println("c1 and c2 are equal");

// but c1 and c2 do compare as "not before and not after" since they represent
the same date and time
if (!c1.after(c2) & !c1.before(c2) {
   System.out.println("c1 and c2 are equivalent");
}
```

Mixing Web services client and server bindings causes errors

Web Services for Java Platform, Enterprise Edition (Java EE) and the Java API for XML-based remote procedure call (JAX-RPC) specifications do not support *round-trip* mapping between Java code and a Web Services Description Language (WSDL) document for all Java types. For example, you cannot turn or serialize a Java Date into XML code and then turn it back or deserialize it into a Java Date. This action deserializes as Java Calendar.

If you have a Java implementation that you create a WSDL document from, and you generate client bindings from the WSDL document, the client classes can be different from the server classes even though the client classes have the same package and class names. The Web service client classes must be kept separate from the Web service server classes. For example, do not place the Web service server bindings classes in a utility Java archive (JAR) file and then include a Web service client JAR file that references the same utility JAR file.

If you do not keep the Web services client and server classes separate, a variety of exceptions can occur, depending on the Java classes used. The following is a sample stack trace error that can occur:

```
com.ibm.ws.webservices.engine.PivotHandlerWrapper TRAS0014I: The following exception was
loggedjava.lang.NoSuchMethodError: com.ibm.wssvt.acme.websvcs.ExtWSPolicyData:
   method getStartDate()Ljava/util/Date;
not found
at com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.addElements(ExtWSPolicyData_Ser.java: 210)
at com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.serialize (ExtWSPolicyData_Swer.java:29)
at com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.serializeActual
(SerializationContextImpl.java 719)
at com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.serialize
(SerializationContextImpl.java: 463)
```

The problem is caused by using an interface as shown in the following example for the service endpoint interface in the service implementation:

```
package server:
public interface Test_SEI extends java.rmi.Remote {
 public java.util.Calendar getCalendar () throws java.rmi.RemoteException;
 public java.util.Date getDate() throws java.rmi.RemoteException;
}
```

When this interface is compiled and run through the **Java2WSDL** command-line tool, the WSDL document maps the methods as shown in the following example:

```
<wsdl:message name="getDateResponse">
 <wsdl:part name="getDateReturn" type="xsd:dateTime"/>
</wsdl:message>

<wsdl:message name="getCalendarResponse">
 <wsdl:part name="getCalendarReturn" type="xsd:dateTime"/>
</wsdl:message>
```

The JAX-RPC mapping implemented by the Java2WSDL tool has mapped both the java.util.Date and java.util.Calendar instances to the xsd:dateTime XML type . The next step is to use the generated WSDL file to create a client for the Web service. When you run the WSDL2Java tool on the generated WSDL, the generated classes include a different version of the server.Test_SEI interface, for example:

```
package server;
public interface Test_SEI extends java.rmi.Remote {
 public java.util.Calendar getCalendar() throws java.rmi.RemoteException;
 public java.util.Date getDate() throws java.rmi.RemoteException;
}
```

The client version of the service.Test_SEI interface is different from the server version in that both the getCalendar and getDate methods return java.util.Calendar. The serialization and deserialization code that the client expects is the client version of the service endpoint interface. If the server version inadvertently is contained in the client CLASSPATH variable, at either compilation or run time, an error occurs.

In addition to theNoSuchMethod error, the IncompatibleClassChangeError and ClassCastException can occur. However, almost any run-time exception can occur. The best practice is to be diligent about separating client Web services bindings classes from server Web services bindings classes. Always place the client bindings classes and server bindings classes in separate modules. If theses binding classes are in the same application, place the bindings classes in utility JAR files that are not shared between modules.

Web services authentication, authorization and secure transport troubleshooting tips

Web services are developed and implemented based on the Web Services for Java Platform, Enterprise Edition (Java EE) specification. The are several troubleshooting authentication and authorization considerations when you are securing Web services.

These Web services are developed and implemented based on the Web Services for Java Platform, Enterprise Edition (Java EE) specification. This topic discusses troubleshooting authentication, authorization, and transport issues to consider when you are securing Web services.

Specifying remote WSDL using HTTPS transport protocol

If your Java API for XML-Based Web Services (JAX-WS) client application specifies a remote address for the WSDL location that requires HTTPS secure communication, and you do not complete the SSL configuration, then an exception occurs. When specifying the WSDL URL using HTTPS transport protocol, you must complete the SSL configuration before the client instance is created. To configure SSL, set the `com.ibm.SSL.ConfigURL` system property as name of the SSL configuration.

The following is an example of a Web services client that specifics the remote WSDL file location using the HTTPS transport protocol:

```
@WebServiceClient(name = "SampleService", targetNamespace = "http://jaxws.sample.websphere.ibm.com/",
  wsdlLocation = "https://localhost:9443/Sample/SampleServicePort?WSDL")
public class SampleService
    extends Service
{

    private final static URL SAMPLESERVICE_WSDL_LOCATION;

    static {
        URL url = null;
        try {
            url = new URL("https://localhost:9080/Sample/SampleService?WSDL");
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        SAMPLESERVICE_WSDL_LOCATION = url;
    }

    ...
}
```

To learn more about setting this system property, read about Setting up the SSL configuration for clients in the ssl.client.props client configuration documentation.

Authentication challenge or authorization failure is displayed

You might encounter an authentication challenge or an authorization failure if a thread switch occurs. For example, an application might create a new thread or a raw socket connection to a servlet might open. A thread switch is not recommended by the Java EE specification because the security context information is stored in thread local. When a thread switch occurs, the authenticated identity is not passed from thread local to the new thread. As a result, WebSphere Application Server considers the identity to be unauthenticated. If you must create a new thread, you must propagate the security context to the new thread. However, this process is not supported by WebSphere Application Server.

Web services security enabled application fails to start

When a Web services security-enabled application fails to start, you might receive an error message similar to the following:

```
[6/19/03 11:13:02:976 EDT] 421fdaa2 KeyStoreKeyLo E WSEC5156E: An exception
while retrieving the key from KeyStore object:
java.security.UnrecoverableKeyException: Given final block not properly padded
```

The cause of the problem is that the keypass value or password provided for a particular key in the key store is invalid. The key store values are specified in the <KeyLocators> elements of one of following binding files: `ws-security.xml`, `ibm-webservices-bnd.xmi` or `ibm-webservicesclient-bnd.xmi`. Verify that the keypass values for keys specified in the <KeyLocators> elements are correct.

Note: Policy sets can only be used with JAX-WS applications. Policy sets cannot be used for JAX-RPC applications.

Applications with Web services security enabled cannot interoperate between WebSphere Application Server Version 6.0.x and Version 5.0.2

Applications with Web services security enabled cannot interoperate between WebSphere Application Server Version 6.0.*x* and Version 5.0.2. When applications attempt to interoperate, a ″digest mismatch″ error is displayed. An error exists in the canonicalization algorithm for XML digital signature, which is fixed in Version 5.1. For Web services security to interoperate between WebSphere Application Server Version 6 and Version 5.0.2, you must update your Version 5.0.2 application server. To update your Version 5.0.2 server, access the WebSphere Application Server Support Web site and download the latest fix pack for WebSphere Application Server, Version 5.0.2.

Application client sending SOAP request receives errors

Use this information to diagnose and troubleshoot problems with clients sending SOAP requests.

What kind of problem are you seeing?

- SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect
- javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria can be found.

If none of these errors match the one you see:

- Browse the application server logs. Look up any error or warning messages in the message table.

  View `*install_dir*/*server_name*/SystemErr.log` and SystemOut.log for clues. See Viewing JVM logs for more information.
- See “Universal Discovery, Description, and Integration, Web service, and SOAP component troubleshooting tips” on page 14 for more information.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, see Troubleshooting help from IBM.

SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect

The most likely cause of this refused connection is that it was sent to the default port, 80, and an HTTP server is not installed or configured.

To verify this situation, send the message directly to the SOAP port; for example, to `http://`*`hostname`*`:9080`. If the message is sent correctly, there are two ways to resolve the problem:

- Continue specifying port 9080 on SOAP requests.
- If an HTTP server is not installed, install one and the associated plug-in component.
- If an HTTP server is installed:
  - Regenerate the HTTP plug-in configuration in the administrative console by clicking **Environment > Update WebServer Plugin**, and restarting the HTTP server.
  - If the problem persists, view the HTTP server access and error logs, as well as the *plugin_install_root*/`logs/`*web_server_name*/`http_plugin.log` file for more information.

javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria can be found

This error usually indicates that new or updated security keys are needed. The security key files are:

- SOAPclient
- SOAPserver
- sslserver.p12

In an installed application, these files are located in the: *`profile_root`*`/installedApps/`*`application_name`*`.ear/soapsec.war/key/` directory.

After replacing these files, you must stop and restart the application. The *`profile_root`* variable refers to the *`profile_root`*`ND/profiles/`*`profile_name`* directory

To replace these files in a SOAP-enabled application that is not yet installed:

- Expand the *application_name*.ear file.
- Expand the *soapsec.war* file.
- Replace the security key files in the `key/` directory.
- After you replace these files, install the application and restart the server.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Universal Discovery, Description, and Integration, Web service, and SOAP component troubleshooting tips

Use this information if you are having problems deploying or running applications that use WebSphere Application Server Web services, Universal Discovery, Description, and Integration (UDDI), or SOAP components.

Try these steps:

- Review the troubleshooting documentation for messaging in the information center:
  - WSIF troubleshooting tips
- Investigate the following areas for SOAP-related problems:

- View the JVM logs for the target application server, and run the Log and Trace Analyzer on the server's service log.
- View the error log of the HTTP server to which the SOAP request is sent.
- View the run-time behavior of the SOAP component in more detail, by enabling trace for `org.apache.soap.*` and `com.ibm.*.soap*`.
- Browse the Web site `http://xml.apache.org/soap/` for FAQs and known SOAP issues.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Tracing Web services

You can trace the Web services runtime components, including an unmanaged client, a managed client and a server application. The procedure entry and exit, as well as the processing actions are traceable in the runtime components. You can also trace user-defined exceptions and SOAP messages that use Java Message Service (JMS) or HTTP to request Web services.

Before you begin

The `com.ibm.ws.webservices.engine.*=all=enabled` specification traces the Web services run time only. See step 4 for settings that you can use to trace user-defined exceptions and SOAP messages or review the tracing SOAP messages with tcpmon documentation to learn about tracing SOAP messages with the tcpmon process.

About this task

The following tasks describe how you can enable trace for Web services:

1. Enable trace for a Web services unmanaged client.
   a. Create a trace properties file by copying the *app_server_root*/properties/TraceSettings.properties file to the same directory as your client application Java archive (JAR) file.
   b. Edit the properties file and change the traceFileName value to output the trace data. For example, traceFileName=/*myDir*/*myAppClient*.trc.
   c. Edit the properties file to remove `com.ibm.ejs.ras.*=all=enabled` and add `com.ibm.ws.webservices.engine.*=all=enabled`.
   d. Add the -DtraceSettingsFile=<*trace_properties_file*> option to the **java** command line that is used to run the client, where *trace_properties_file* represents the name of the properties file that you created in the substeps a through c. For example, `java -DtraceSettingsFile=TraceSettings.properties myApp.myAppMainClass`.
2. Enable trace for a Web services-managed client by invoking the **launchClient** command-line tool with the following options:

   -CCtrace=com.ibm.ws.webservices.engine.*=all=enabled

   -CCtracefile=traceFileName For example:

   ```
   app_server_root/bin/launchClient MyAppClient.ear
   -CCtrace=com.ibm.ws.webservices.engine.*=all=enabled -CCtracefile=myAppClient.trc
   ```

See launchClient tool for more information.

3. Enable trace for a Web Services for Java Platform, Enterprise Edition (Java EE) server application.
   a. Start WebSphere Application Server.
   b. Open the administrative console.
   c. Click **Servers** > **Application Servers** > *server*.
   d. Click **Change Log Detail Levels**.
   e. Add or delete the trace string in the text box. For this task, delete the trace string `*=info` and add the trace string `com.ibm.ws.webservices.engine.*=all=enabled`. You can specify the trace string in the text box in one of two ways:
      - Type the trace string directly into the text box. You must separate each trace string by a colon (:) with no spaces. For example:
        ```
        com.ibm.ws.webservices.trace.MessageTrace=finest:com.ibm.ws.webservices.
        engine.Message=finest
        ```
      - Choose a predefined trace string from the section that is listed. The predefined section starts with `*[All Components]`. The predefined tracing strings Web services component are listed under the com.ibm..ws.* section.
        - Click the plus (+) sign to expand the **com.ibm.ws.*** section.
        - Click the predefined trace string. For example, if you want to add a predefined trace string for the SOAP messaging trace, you can click: `com.ibm.ws.webservices.trace.MessageTrace`.
        - Click the trace option from the drop-down list. For example, you can choose off, fatal, severe, warning, audit, info, config, detail, fine, finer, finest, and all. The option, finest, is recommended. When you click on the option, the option is added to the end of the trace string. For example:
          ```
          com.ibm.ws.webservices.trace.MessageTrace=finest
          ```
   f. Click **Save** and **Apply**.

   For more information see, Enabling trace.
4. Enable trace for SOAP messages, user-defined exceptions, or both. The following trace specifications are used to trace SOAP messages:
   - `com.ibm.ws.webservices.trace.MessageTrace=all`

     This specification traces the contents of a SOAP message, including the binary attachment data.

     When the context-type of the SOAP message is not text and xml, the message probably contains attachments. In this case, the message is displayed in the trace file in the hex dump format. The following example illustrates a line in the hex dump format for non-text SOAP messages:
     ```
     0000: 0D 0A 2D 2D 2D 2D 2D 2D - 3D 5F 50 61 72 74 5F 36 ..------=_Part_6
     ```
     - In each trace file line, 16 bytes of the message are displayed
     - The first four digits are a hex number whose value is the byte offset into the SOAP message of the first byte on the line.
     - The next 16 two-digit hex numbers are the contents of each of the consecutive bytes in the message.
     - The ASCII representation of the bytes is displayed in the last 16 characters of the line, with unprintable characters that are represented by a period.
   - `*=off:com.ibm.ws.webservices.*=all`

     You can trace all Web services information, including the SOAP messages and the user-defined exceptions, with this setting.

   You can enable logging of user-defined exceptions by specifying the `com.ibm.ws.webservices.trace.UserExceptionTrace=all` trace string. The user-defined exceptions are not logged by default. A user-defined exception is an exception that is defined in the Web Services Description Language (WSDL) file for an operation.

A user-defined exception often indicates an error-free condition. For example, the user-defined OverdrawnException exception, can occur for the service endpoint implementation of the makeWithdrawl method. This exception indicates an expected condition and does not indicate an error in the service endpoint implementation. Because these types of exceptions can occur during normal processing, they are not logged by default. When a user-defined exception is logged, the information is sent to the `trace.log` file and not to the `SystemOut.log` file.

You can also use the following trace strings to enable tracing for user-defined exceptions, as well as other trace points:

- com.ibm.ws.webservices.*=all

  Turns on all Web services run-time trace logs.
- com.ibm.ws.webservices.trace.*=all

  Turns on MessageTrace and UserExceptionTrace.

Results

You have enabled trace for the unmanaged clients, managed clients, and the server applications. Depending on the trace string specification, the trace can include run-time components, user-defined exceptions and SOAP messages.

What to do next

Analyze the message data.

Tracing SOAP messages with tcpmon

You can trace SOAP messages that request Web services by using the tcpmon tool.

Before you begin

You can use other trace tools to trace SOAP messages, similar to how you can trace Web services components. For further information, see the Tracing Web services chapter in the *Administering applications and their environment* PDF book.

It is not recommended that you use the tcpmon tool in a stressed environment. Tcpmon is only for monitoring SOAP messages in a lightweight environment.

About this task

You can trace SOAP messages exchanged between a client and the server by installing a monitor or sniffer application to capture the HTTP traffic between the two points. The application server product provides a utility class, `com.ibm.ws.webservices.engine.utils.tcpmon`, to trace the SOAP messages. The `com.ibm.ws.webservices.engine.utils.tcpmon` class redirects messages from a port, records the messages, and forwards the messages to another port.

WebSphere Application Server typically listens on port 9080, or port 80 if you are using IBM HTTP Server. The tcpmon process can be configured to listen on a particular port, such as 9088, while redirecting messages to another port, such as 9080 or port 80. The client is redirected to use port 9088 to access Web services.

Redirecting an application client to a different port is done by changing the SOAP address in the client Web Services Description Language (WSDL) file to use port 9088 and then running the **wsdeploy** command-line tool on the client enterprise archive (EAR) file to regenerate the service implementation.

The wsdeploy command is supported by Java API for XML-based RPC (JAX-RPC) applications. The Java API for XML-Based Web Services (JAX-WS) programming model that is implemented by the application server does not support the wsdeploy command. If your Web services application contains only JAX-WS endpoints, you do not need to run the wsdeploy command, as this command is used to process only JAX-RPC endpoints.

1. Run the following command to display a window labeled TCPMonitor:

```
java -cp app_server_root/runtimes/com.ibm.ws.webservices.thinclient_7.0.0.jar com.ibm.ws.webservices.engine.utils.tcpmon
```

2. Configure the TCPMonitor to listen on port 9088 and forward messages to port 9080.
   a. In the Listen Port # field, enter `9088`.
   b. Click **Listener**.
   c. In the TargetHostname field, enter `localhost`.
   d. In the Target Port # field, enter `9080`.
   e. Click **Add**.
   f. Click the **Port 9088** tab that displays on the top of the page.

Results

The messages exchanged between the client and server display in the TCPMonitor Request and Response pane.

What to do next

Save the message data and analyze it.

Frequently asked questions about Web services

This topic presents frequently asked questions about the development and implementation of Web services.

- What is the relationship of the WebSphere product to Apache open source ?
- What IBM development tools work with Web Services?
- Is Web Services for Java EE technology part of the Java EE specification?
- What is the relationship between Apache SOAP 2.3 and the Web Services for Java EE specification?
- What standards does the Web Services for Java EE component of WebSphere Application Server support?
- Does the Web Services for Java EE technology interoperate with other SOAP implementations, like .NET?
- Can I use a JavaBeans component to implement a Web service using SOAP over Java Message Service (JMS) invocation?
- Does the SOAP over JMS support interoperate with other vendors?
- How does two-way messaging with a SOAP over JMS implementation work? Can it support multiple clients making simultaneous requests?

What is the relationship of the WebSphere product to Apache open source?

The WebSphere product has always extensively supported open source. From a Web services perspective, the WebSphere product contributes a large percentage of the JAX-RPC specification to the open source Apache Axis community. WebSphere Community Edition uses the Apache Axis runtime for its support for JAX-RPC 1.1. With the movement of Web services to a more messaging-centric asynchronous model, the Apache Axis community has created a new version of a Web services runtime that is based on the StAX architecture entitled Axis2.

Axis2 introduces its own proprietary programming and deployment model that is agnostic of any Java-based JCP standards. It did this primarily so that it could support multiple Java-based programming models, whether it JAX-WS, SCA (Apache Tuscany) and Groovy. While the application server implements a standards-based JAX-WS programming model, it actually uses a version of Axis2 as part of its implementation. You might see messages during tracing or within call stacks that reflect its Axis2 origins.

The WebSphere product is supporting only the JAX-WS programming model and the deployment model that is documented in the information center. Any usage of native Axis2 APIs is not supported by the WebSphere product.

What IBM development tools work with Web Services?

The Rational® Application Developer assembly tools provide a graphical interface for developing code artifacts, assembling the code artifacts into various archives or modules, and configuring related Java EE deployment descriptors.

Is Web Services for Java EE technology part of the Java EE specification?

WebSphere Application Server Version 7.0 and later is based on Web Services for Java Platform, Enterprise Edition (Java EE) 5. Prior to Java EE 5, the specification name was Java 2 Platform, Enterprise Edition (J2EE). WebSphere Application Server Version 6.x is based on J2EE 1.4. For WebSphere Application Server Version 5.0.2 and Version 5.1.x, the Web Services for J2EE Version 1.0 specification is an addition to J2EE 1.3. The J2EE specification 1.4 requires support for Web Services for J2EE Version 1.1. Minor differences exist between the J2EE 1.3 Version (JSR-109 Version 1.0) and the J2EE 1.4 Version (JSR-109 Version 1.1).

What is the relationship between Apache SOAP 2.3 and the Web Services for Java EE specification?

The development and implementation of a Web service is based on the Web Services for J2EE specification in Version 6.0.x and later. You are encouraged to migrate from Apache SOAP because this approach is not recommended for future releases. For information about migrating your Apache SOAP Web services, see Migrating Apache SOAP Web services to JAX-RPC Web Services based on Java EE standards.

What standards does the Web services run time support?

You can review the standards and specifications that are supported by WebSphere Application Server for the Web services run time in Specifications and API documentation.

Does the Web Services for Java EE technology interoperate with other SOAP implementations, like .NET?

WebSphere Application Server supports Web services that are consistent with the WS-I Basic Profile, and should interoperate with any other vendor conforming to this specification.

Can I use a JavaBeans component to implement a Web service using SOAP over Java Message Service (JMS) invocation?

The SOAP over JMS support provides access only to enterprise beans-based Web services If you want to use a JavaBeans implementation instead of an enterprise bean to implement the service endpoint, you must create a *facade* enterprise bean that delegates to the JavaBeans implementation.

Does the SOAP over JMS support interoperate with other vendors?

Before WebSphere Application Server Version 7.0, no specification has existed that describes interoperability requirements for SOAP over JMS implementations. WebSphere Application Server Version 7.0 introduces support for the emerging industry standard SOAP over Java Message Service specification. This proposed standard provides a standard set of interoperability guidelines for using a JMS-compliant transport with SOAP messages to enable interoperability between the implementations of different vendors. Support for this emerging standard positions WebSphere to be able to interoperate with other vendor implementations of SOAP over JMS as this standard is adopted. While the specification is in draft form and not yet final, WebSphere Application Server Version 7.0 supports the current SOAP over JMS draft specification. To learn more about this specification, see the specifications and API documentation.

How does two-way messaging with a SOAP and JMS implementation work? Can it support multiple clients making simultaneous requests?

When using two-way Web services operations, the client can choose to use a permanent reply queue or the Web services run time will, by default, use a temporary JMS queue. When the client issues a two-way request, the underlying Web services run time creates a temporary JMS queue, if a permanent queue is not being used, to receive the response. The reply queue, either temporary or permanent, is specified as the replyTo destination that is in the outgoing JMS request message. After the server processes the request, it directs the response to the replyTo destination specified in the request message. The client deletes the temporary queue, if a permanent queue was not used, after the response is received. The server can handle simultaneous requests from multiple clients because each incoming request message contains the destination to which the reply is sent.

Web services security troubleshooting tips

To troubleshoot Web services security, review the configurations with assembly tools to match the client and server request and the response configurations.

Troubleshooting Web services security is best done by reviewing the configurations with assembly tools so that you can match up the client and server request and the response configurations. These configurations must match. A client request sender configuration must match a server request receiver configuration. For encryption to successfully occur, the public key of the receiver must be exported to the sender and this key must be configured properly in the encryption information. For authentication, you must specify the method used by the client in the login mapping of the server.

For more information about the assembly tools, read the assembly tools section of the *Developing and deploying applications* PDF book.

The following includes a list of generic troubleshooting steps that you can perform.

Steps for this task

1. Verify that the client security extensions and server security extensions match on each downstream call for the following senders and receivers:
   - Request sender and request receiver
   - Response sender and response receiver
2. Verify that when the **Add Created Time Stamp** option is enabled on the client-side that the server has the **Add Received Time Stamp** option configured. You must configure the security extensions with an assembly tool.
3. Verify that the client security bindings and the server security bindings are correctly configured. When the client authentication method is signature, make sure that the server has a login mapping. When the client uses the public key `cn=Bob,o=IBM,c=US` to encrypt the body, verify that this Subject is a personal certificate in the server key store so that it can decrypt the body with the private key. You can configure the security bindings using an assembly tool or the WebSphere Application Server administrative console.
4. Check the `SystemOut.log` file in the `${`*`USER_INSTALL_ROOT`*`}/logs/`*`server1`* directory (*server1* changes depending upon the server name) for messages that might provide information about the problem.
5. Enable trace for Web services security by using the following trace specification:

   ```
   com.ibm.xml.soapsec.*=all=enabled:com.ibm.ws.webservices.*=all=enabled:
   com.ibm.wsspi.wssecurity.*=all=enabled:com.ibm.ws.security.*=all=enabled: SASRas=all=enabled
   ```

   Type the previous three lines as one continuous line.

Errors when securing Web services

The following errors might occur when you secure Web services:

- “″CWWSI5061E: The SOAP Body is not signed″ error message displays”
- “″CWWSI5075E: No security token found that satisfies any one of the authentication methods″ error message displays” on page 22
- “″CWWSI5094E: No UsernameToken of trusted user was found or the login failed for the user while the TrustMode is BasicAuth″ error message displays” on page 23
- “″CWSCJ0053E: Authorization failed for /UNAUTHENTICATED...″ error message displays” on page 23
- “″WSWS3243I: Info: Mapping Exception to WebServicesFault.″ error message is displayed when you specify the value type local name and the URI for a token consumer or the token generator” on page 24
- “″Invalid URI: The format of the URI could not be determined″ error message might display when you use a Microsoft .NET client that accesses a Web service for WebSphere Application Server” on page 24
- “″WSEC5502E: Unexpected element as the target element″ error message displays ” on page 25
- “″WSEC6664E: Null is not allowed to PKIXBuilderParameters. The configuration of TrustAnchor and CertStoreList are not correct″ exception displays” on page 25
- “″WSE567: The incoming Username token must contain both a nonce and a creation time for the replay detection feature″ Microsoft .NET error displays” on page 26
- “″WSEC6500E: There is no candidate used to login″ error message displays” on page 29
- “Instead of issuing a CertPath exception, a valid certification path is built on Sun Solaris when an invalid certificate is used” on page 30
- “Hardware cryptographic requests with card-related exceptions must use cryptographic software to complete requests successfully” on page 31

Version 5.x application

″CWWSI5061E: The SOAP Body is not signed″ error message displays

Cause:

This error usually occurs whenever the SOAP security handler does not load properly, and does not sign the SOAP body. The SOAP security handler is typically the first validation that occurs on the server-side, so many problems can cause this message to display. The error might be caused by invalid actor URI configurations.

Solution:

You can configure the actor Universal Resource Identifier (URI) at the following locations within the assembly tool:

- From the Web services client editor within the assembly tool for client configurations:
  - Click **Security Extensions > Client Service Configuration Details** and indicate the actor information in the **ActorURI** field.
  - Click **Security Extensions > Request Sender Configuration section > Details** and indicate the actor information in the **Actor** field.
- From the Web Services Editor within the assembly tool for server configurations:
  - Click **Security Extensions > Server Service Configuration** section. Verify that the actor URI has the same actor string as the client-side.
  - Click **Security Extensions > Response Sender Service Configuration Details > Details** and indicate the actor information in the **Actor** field.

The actor information on both the client and the server must refer to the same string. When the actor fields on the client and the server match, the request or response is acted upon instead of being forwarded downstream. The actor fields might be different when you have Web services acting as a gateway to other Web services. However, in all other cases, verify that the actor information matches on the client and server. When the Web services implementation is acting as a gateway and it does not have the same actor configured as the request passing through the gateway, this Web services implementation does not process the message from the client. Instead, it sends the request downstream. The downstream process that contains the correct actor string processes the request. The same situation occurs for the response. Therefore, it is important that you verify that the appropriate client and server actor fields are synchronized.

Additionally, the error can appear when you do not specify that the body is signed in the client configuration. To sign the body part of the message using the Web service client editor in the assembly tool, click **Security Extensions > Request Sender Configuration > Integrity** and select the message parts to sign.

Version 5.x application

″CWWSI5075E: No security token found that satisfies any one of the authentication methods″ error message displays

Solution:

Verify that the client and server login configuration information matches in the security extensions. Also, verify that the client has a valid login binding and that the server has a valid login mapping in the security bindings. You can check this information by looking at the following locations in the assembly tool:

- From the Web services client editor within the assembly tool for client configurations:
  - Click **Security Extensions > Request Sender Configuration > Login Configuration** verify the authentication method.
  - Click **Port Binding > Security Request Sender Binding Configuration > Login Binding** verify the authentication method and other parameters.
- From the Web Services Editor within the assembly tool for server configurations:
  - Click **Security Extensions > Request Receiver Service Configuration Details > Login Configuration** and verify the authentication method.
  - Click **Binding Configurations > Request Receiver Binding Configuration Details > Login Mapping** and verify the authentication method and other parameters.

Also, make sure that the actor URI specified on the client and server matches. You can configure the actor URI at the following locations within the assembly tool:

- From the Web services client editor within the assembly tool for client configurations:
  - Click **Security Extensions > Client Service Configuration Details** and indicate the actor information in the **ActorURI** field.
  - Click **Security Extensions > Request Sender Configuration section > Details** and indicate the actor information in the **Actor** field.
- From the Web services editor within the assembly tool for server configurations:
  - Click **Security Extensions > Server Service Configuration** section. Make sure that the **Actor URI** field has the same actor string as the client side.
  - Click **Security Extensions > Response Sender Service Configuration Details > Details** and indicate the actor information in the **Actor** field.

″CWWSI5094E: No UsernameToken of trusted user was found or the login failed for the user while the TrustMode is BasicAuth″ error message displays

Cause:

This situation occurs when you have IDAssertion configured in the login configuration as the authentication method.

Solution:

On the sending Web service, configure a trusted basic authentication entry in the login binding. Then, on the server side, verify that the trusted ID evaluator has a property set that contains the user name of this basic authentication entry.

To configure the client for identity assertion, consult the following topics:

- Configuring the client for identity assertion: specifying the method
- Configuring the client for identity assertion: collecting the authentication method

To configure the server for identity assertion, consult the following topics:

- Configuring the server to handle identity assertion authentication
- Configuring the server to validate identity assertion authentication information

″CWSCJ0053E: Authorization failed for /UNAUTHENTICATED...″ error message displays

Cause:

The following authorization error occurs with UNAUTHENTICATED as the security name: `CWSCJ0053E: Authorization failed for /UNAUTHENTICATED while invoking (Home)com/ibm/wssvt/tc/pli/ejb/ Beneficiary findBeneficiaryBySsNo(java.lang.String):2 securityName: /UNAUTHENTICATED;accessID: null is not granted any of the required roles: AgentRole`

This situation occurs because a login configuration is not being configured or Web services Security is not configured from a client to a server. When the request arrives at the server and authentication information is not received, the UNAUTHENTICATED user is set on the thread. Authorization returns this error if there are any roles assigned to the resource except for the special ″Everyone″ role, which supports access by anyone.

If the client successfully authenticates to an Enterprise JavaBeans (EJB) file, but the EJB file calls a downstream EJB file that is not configured with Web services security or transport security, such as HTTP user ID and password, an error can occur for this downstream request.

Solution:

Using the assembly tool, verify that the enterprise archive (EAR) file for both client and server has the correct security extensions and security bindings. For more information, consult the following topics:

- Configuring the client security bindings using an assembly tool
- Configuring the security bindings on a server acting as a client using the administrative console
- Configuring the server security bindings using an assembly tool
- Configuring the server security bindings using the administrative console

″WSWS3243I: Info: Mapping Exception to WebServicesFault.″ error message is displayed when you specify the value type local name and the URI for a token consumer or the token generator

Cause:

The Value type URI is not required for the following predefined value type local names:

- Username token
- X509 certificate token
- X509 certificates in a PKIPath
- A list of X509 certificates and CRLs in a PKCS#7

Solution:

If you specify one of the previous value type local names, do not enter a value for the Value type URI field.

″Invalid URI: The format of the URI could not be determined″ error message might display when you use a Microsoft® .NET client that accesses a Web service for WebSphere Application Server

Cause:

The following exception message might display when you use a Microsoft .NET client that accesses a Web service for WebSphere Application Server.

```
Invalid URI: The format of the URI could not be determined.
```

Exception type:

```
System.UriFormatException
at System.Uri.Parse()
at System.Uri..ctor(String uriString, Boolean dontEscape)
at System.Uri..ctor(String uriString)
at Microsoft.Web.Services2.SoapInputFilter.CanProcessHeader(XmlElement header, SoapContext context)
at Microsoft.Web.Services2.Security.SecurityInputFilter.ProcessMessage(SoapEnvelope envelope)
at Microsoft.Web.Services2.Pipeline.ProcessInputMessage(SoapEnvelope envelope)
at Microsoft.Web.Services2.InputStream.GetRawContent()
at Microsoft.Web.Services2.InputStream.get_Length()
at System.Xml.XmlScanner..ctor(TextReader reader, XmlNameTable ntable)
at System.Xml.XmlTextReader..ctor(String url, TextReader input, XmlNameTable nt)
at System.Xml.XmlTextReader..ctor(TextReader input)
at System.Web.Services.Protocols.SoapHttpClientProtocol.ReadResponse(SoapClientMessage message,
WebResponse response, Stream responseStream, Boolean asyncCall)
```

Within WebSphere Application Server, Web services security is enabled and uses the ActorURI attribute. This error occurs because Microsoft .NET Web Services Enhancements (WSE) Version 2.0 Service Pack 3 does not support a relative URI value for the ActorURI attribute. WSE Version 2.0 Service Pack 3 supports an absolute Uniform Resource Identifier (URI) for this attribute only.

Solution:

To work with a Microsoft .NET client, you must configure this attribute as an absolute URI. An example of an absolute URI is: abc://myWebService. An example of a relative URI is: myWebService.

To configure ActorURI attribute for use with WebSphere Application Server, use the IBM assembly tool to complete the following steps:

1. Open the Web Service Editor, click the **Extensions** tab and expand **Server Service Configuration**.
2. Enter the full absolute URI in the Actor field.
3. Expand **Response Generator Service Configuration Details > Details**.
4. Enter the full absolute URI in the Actor field.

For more information on assembly tools, see the topic Assembly tools.

″WSEC5502E: Unexpected element as the target element″ error message displays

Cause:

If the following error displays, the cause may be an X.509 token that is in a message, but doesn't have a matching token consumer configured. This error can occur on either consumer or provider JAX-RPC applications.

```
com.ibm.wsspi.wssecurity.SoapSecurityException: WSEC5502E: Unexpected element as the target element: wsse:BinarySecurityToken
```

The cause of this error is that either an X.509 token is configured, and an X.509v3 token is received, or an X.509v3 token is configured, and an X.509 token is received. This happens most often when receiving X.509v3 tokens from Microsoft .NET. To determine if this is the cause of the problem, follow these steps:

1. Obtain a WS-Security trace for the process that is producing the message. For more information on how to implement the WS-Security trace, read the topic "Tracing Web services" on page 15.
2. Check to see if the trace contains information about the incoming SOAP message:
   a. From the point of the exception, search backwards for the term **soapenv:env**.
   b. From that point, search backwards for the term **X509**.
   c. Note the type of the X.509 security token, either **#X509** or **#X509v3**.
3. If the trace does not contain information about the incoming SOAP message, for example, because the trace is incomplete, search backwards for the term **Target's value type is**, starting at the point of the exception. This search locates the part of the trace that shows which security token was being processed at the time of the error. Note the type of the security token, either #X509 or #X509v3.
4. Check the type of X.509 security token that is specified in the consumer configuration:
   a. From the point of the exception, search backwards for the term **WSSConsumerConfig**.
   b. Now search forward for the term **#X509**.
   c. Note the type of the configured X.509 security token consumer, either **#X509** or **#X509v3**.
5. If the configured token consumer does not match the type of the incoming security token, then this confirms that a security token type mismatch is the cause of the error.

Solution:

The configured token consumer must match the type as specified for the inbound security token. If the cause of the error, as determined in the previous steps, is determined to be a security token type mismatch, then you must change either the consumer or the provider configuration for WS-Security to ensure that the token types match.

″WSEC6664E: Null is not allowed to PKIXBuilderParameters. The configuration of TrustAnchor and CertStoreList are not correct″ exception displays

Cause:

The certificate path setting is not configured properly.

Solution:

Configure the certificate path setting by completing the following steps:

1. In the administrative console, click **Security > Web services**.
2. Under the Default consumer binding heading, click **Signing information > *configuration_name***.
3. Select either the **Trust any** or **Dedicated signing information** option.

If you select the **Dedicated signing information** option, select both a trust anchor and a certificate store from the configurations that are provided in the drop-down lists.

4. Click **OK** and **Save** to the master configuration.

″WSE567: The incoming Username token must contain both a nonce and a creation time for the replay detection feature″ Microsoft .NET error displays

Cause:

In this scenario, you have a Web services client for WebSphere Application Server and a Microsoft .NET Web service. The Microsoft .NET Web service has a ws-security constraint for a username token configured. The following exception is thrown from the Microsoft .NET server:

WSE567: The incoming username token must contain both a nonce and a creation time for the replay detection feature.

By default, the Microsoft .NET Web service validates the nonce and the timestamp for the username token. However, it is optional for you to configure the nonce and timestamp properties for a Web service client that is using WebSphere Application Server.

Solution:

Complete the following steps to add the nonce and timestamp properties for a username token on a Web service client for WebSphere Application Server. These steps involve an assembly tool. For more information about assembly tools, read the assembly tools section of the *Developing and deploying applications* PDF book. Information is also available in the topic Assembly tools.

1. Open the Web service client deployment descriptor and click the **WS-Binding** tab.
2. Expand the Security Request Generator Binding Configuration > Token Generator sections.
3. Click the name of the username token that you already created and click **Edit**.
4. In the Properties section of the Token Generator window, click **Add**.
5. Enter `com.ibm.wsspi.wssecurity.token.username.addNonce` in the Name field to provide the name of the nonce property.
6. Enter `true` in the **Value** field.
7. Click **Add**.
8. Enter `com.ibm.wsspi.wssecurity.token.username.addTimestamp` in the Name field to provide the name of the timestamp property.
9. In the Value field, enter `true`.
10. Click **OK** and save the client deployment descriptor.

Java 2 Security exceptions occur when using the com.ibm.wsspi.wssecurity.auth.token package with Java 2 Security enabled

Cause:

An application creates Java 2 Security exceptions while using the com.ibm.wsspi.wssecurity.auth.token.* package when Java 2 Security is enabled.

New Java 2 permissions have been set for various public methods of the com.ibm.wsspi.wssecurity.auth.token.* package on WebSphere Application Server Version 6.1. If your application uses one of the public methods from these classes that are protected by Java 2 Security permissions and it does not have the appropriate permissions, the application will fail. The exception message provides information that identifies the classes and public methods that are affected with the corresponding new Java 2 Security permission.

Solution:

Grant permission in the was.policy file for the application:

1. Use the PolicyTool to edit the policy files. Follow the appropriate steps for your operating system.
2. Add all of the permissions to the was.policy file that gets packaged in the enterprise archive (EAR) file for your application. If you want finer granularity for the permissions in the was.policy file for your application, enable the permissions that are necessary for your application based upon the classes that you need.

   For example, if you need to access only the methods for the X509BSToken, you would add the following permissions to the was.policy file:

```
grant codeBase "file:${application}" {
   permission com.ibm.websphere.security.WebSphereRuntimePermission
     "wssecurity.X509BSToken.setBytes";
   permission com.ibm.websphere.security.WebSphereRuntimePermission
     "wssecurity.X509BSToken.setCert";
  permission com.ibm.websphere.security.WebSphereRuntimePermission
     "wssecurity.WSSToken.setTrusted";
  permission com.ibm.websphere.security.WebSphereRuntimePermission
     "wssecurity.WSSToken.addAttribute";
  permission com.ibm.websphere.security.WebSphereRuntimePermission
     "wssecurity.WSSToken.setUsedTokenConsumer";
};
```

3. Update the was.policy file in the EAR file for your application.
4. Uninstall the application from WebSphere Application Server and reinstall it with the new EAR file and the updated was.policy file.

An exception might occur when integrity or confidentiality is asserted for a SOAP element

Cause:

If a client asserts integrity or confidentiality for a SOAP element but the element is missing from the message, an exception is issued.

If the client requires that the application of a signature or encryption to a SOAP element, the SOAP element must always be present. The presence of this element is not optional. For example, if the configuration specifies that integrity or confidentiality must be applied to the wscontext element. If wscontext is missing from the message, the following exception is issued:

```
com.ibm.wsspi.wssecurity.SoapSecurityException: WSEC5636W: Objects
to be processed not found with the dialect
[http://www.ibm.com/websphere/webservices/wssecurity/dialect-was]
and the keyword [wscontext]
```

Solution:

Client developers must assure that the SOAP elements they target for integrity or confidentiality are always present in the SOAP message. If you cannot assure that the SOAP element is always present, do not target the SOAP elements for integrity or confidentiality.

Client output exceptions caused by the difference in JSR-101 and JSR-109 programming models

Cause:

Sometimes client output exceptions are produced when running the client. The client output exceptions might be caused by the differences in the JSR-101 versus JSR-109 programming models.

You can configure any of the Web services security constraints, such as: username token, X509 token, signing or encrypting the SOAP elements, and so on. For example, you can configure the username token on a WebSphere Application Server client and service. The client is configured to send a username token in the request, and the server is configured to expect a username token. But if the client does not send a username token, the server rejects the request. When the client does not perform a Java Naming and Directory Interface (JNDI) naming lookup, the client is probably not a JSR-109 client. If it is not a JSR-109 client, the client will not get the JSR-109 configuration information, including the security configurations, and the request will fail.

For the JSR-109 programming model, the invocation begins with the JNDI lookup, which allows the security configuration information to be attached. For the JSR-101 programming model, a JNDI lookup is not performed; the security configuration information cannot be attached.

Solution:

This behavior is not a problem with the JSR-101 and JSR-109 programming models. Web services security does not provide the WebSphere Application Server security configuration information to a JSR-101 client.

The Web services security implementation works according to the following guidelines:

- Web services security is not supported for a JSR-101 client.
- You can only configure a JSR-109 client to use Web services security.

If the client requires Web services security, it must be a JSR-109 client.

″WSSecurityCon E WSEC5514E: An exception while processing WS-Security message″ error displays

Cause:

The managed client has no access to the Web services deployment descriptor because the lookup() call did not use the Java Naming and Directory Interface (JNDI). Without the lookup() call, the client cannot access the deployment descriptor. The Web services security configuration is in the Web services deployment descriptor. The following is a detail exception that is created:

```
00000046 WebServicesFa 1
com.ibm.ws.webservices.engine.WebServicesFault makeUserFault
MakeUserFault:     com.ibm.wsspi.wssecurity.SoapSecurityException:
WSEC5720E: A required message part [body] is not signed.
 at com.ibm.wsspi.wssecurity.SoapSecurityException.format(SoapSecurityException.java:143)
 at com.ibm.ws.webservices.wssecurity.dsig.VerifiedPartChecker.invoke(VerifiedPartChecker.java:
263)
 at com.ibm.ws.webservices.wssecurity.core.WSSConsumer.checkRequiredIntegrity(WSSConsumer.java:
1430)
 at com.ibm.ws.webservices.wssecurity.core.WSSConsumer.invoke(WSSConsumer.java:545)
 at com.ibm.ws.webservices.wssecurity.handler.WSSecurityConsumerBase.invoke(WSSecurityConsumerB
ase.java:85)
 at com.ibm.ws.webservices.wssecurity.handler.GlobalSecurityHandler.handleRequest6(GlobalSecuri
tyHandler.java:406)
```

Solution:

For the managed clients, the service lookup is through Java Naming and Directory Interface (JNDI) lookup. Read about setting up a UsernameToken, Web services security, digital signature Web services security and Lightweight Third-Party Authentication (LTPA) token Web services security. The following is an example of a context lookup that is JSR 109 compliant:

```
InitialContext ctx = new InitialContext();
FredsBankServiceLocator locator
    =(FredsBankService)ctx.lookup("java:comp/env/service/FredsBankService");
FredsBank fb = locator.getFredsBank(url);
long balance = fb.getBalance();
```

When you are instantiating a context lookup for a managed client, do not use new() for the service locator. Here is an example that is not JSR 109 compliant (new ServiceLocator):

```
Properties prop = new Properties();
InitialContext ctx = new InitialContext(prop);
FredsBankServiceLocator locator = new FredsBankServiceLocator();
FredsBank fb = locator.getFredsBank(url);
long balance = fb.getBalance();
```

″WSEC6500E: There is no candidate used to login″ error message displays

Cause:

This situation can occur in one of the following conditions:

- Application security is enabled, but the inbound SOAP message does not contain the required security token specified in the consumer caller part for the service.
- A Web service client is invoking Web Services by using Web Services Security and application security is disabled on the application server that is hosting the Web service.

For example, a Web service might be configured for authentication by using a Username token or an LTPA token. However, it is deployed to an application server where global security is disabled. When the Web service is invoked by a Web service client, which correctly provides the required Username token or LTPA token, the Web service invocation will fail. The following fault might be returned back to the Web service client:

```
<soapenv:Fault>
<faultcode xmlns:p55="http://docs.oasis-open.org/wss/2004/01/oasis-
      200401-wss-wssecurity-secext-1.0.xsd">p55: FailedAuthentication
</faultcode>
<faultstring>
<![CDATA[com.ibm.wsspi.wssecurity.SoapSecurityException:
WSEC6500E: There is no candidate used to login.]]>
</faultstring>
<detail encodingStyle=""/>
</soapenv:Fault>
```

Solution:

If application security is enabled on the application server that is hosting the Web service, ensure that the client is properly configured to send the security token that is required by the Web service in the Web Services Security consumer caller part configuration.

If application security is not enabled on the application server that is hosting the Web service, do one of the following:

- Enable application security on the application server that is hosting the Web service.
- Set the com.ibm.wsspi.wssecurity.config.disableWSSIfApplicationSecurityDisabled Web services security custom property on the Web service.

The com.ibm.wsspi.wssecurity.config.disableWSSIfApplicationSecurityDisabled property enables Web services security to not process the WS-Security header if application security is disabled. This allows system administrators and application programmers to debug aspects of their services in a non-secure environment without having to remove the WS-Security information from their deployment descriptors. The use of this property is only intended for diagnostic purposes and not for a production environment.

Valid values for this property are **true** and **false**. The default value is **false**.

Application-level, cell level and server-level are the levels of bindings that WebSphere Application Server offers.

To configure the server-level bindings, which are the defaults:

1. Click **Servers** > **Application servers** > ***<server_name>***.
2. Under Security, click **Web Services: Default bindings for Web services security**.
3. Do one of the following:

- Click **Default consumer bindings** > **Properties** (might apply to all applications in the cell).
- Click **Additional Properties** > **Properties** (might apply to all applications in the cell).

To configure the cell-level bindings and access the default bindings on the cell level:

1. Click **Security** > **Web services**.
2. Under Default generator bindings or Default consumer bindings, click **Properties**.
3. Under Security, click **Web Services: Default bindings for Web services security**.
4. Do one of the following, specified in the following locations, in priority order:
   - Click **Default consumer bindings** > **Properties**.
   - Click **Additional Properties** > **Properties**.

To configure a JVM System property:

1. Click **Servers** > **Application servers** > ***<server_name>***.
2. Under Server Infrastructure, click **Java and Process Management** > **Process Definition**.
3. Under Additional Properties, click **Java Virtual Machine**.
4. Under Additional Properties, click **Custom Properties**.

Instead of issuing a CertPath exception, a valid certification path is built on Sun Solaris when an invalid certificate is used

Cause:

WS-Security enabled Web services applications on a Sun Solaris system may incorrectly build a valid certification path even though an invalid certificate has been used. When the key in the certificate in a request and the key in the certificate retrieved on the server do not match, an error message should be issued. However, when the certificates differ in every respect except for the DN, and the Sun security provider is used, the certification path is returned as valid. This problem does not occur when the IBMCertPath security provider is used. IBMCertPath is the default security provider on all systems except Sun Solaris, therefore Sun Solaris is the only system on which this problem occurs.

- When the Web service is deployed to non-Sun Solaris systems, the IBM default CertPath provider (IBMCertPath) is returned. When the code is working properly, you will see the following exception because the keys do not match:

```
WSEC5085E: Unable to build a valid CertPath:  java.security.cert.CertPathBuilderException:  unable to find valid certification path to requested target
```

- When the Web service is deployed to Sun Solaris, the java.security.cert.CertPathBuilder.build method returns the Sun default CertPath provider instead of IBMCertPath. The Sun security provider checks only the distinguished name (DN) and does not check the signature information.

  The request should fail because of the incorrect key. However, the invalid CertPath is returned as valid because only the DN was checked. Web services applications running on Sun Solaris could incorrectly build a valid CertPath if given invalid input that is different in every respect from a certificate on the server side, except for the DN.

Solution:

A new property has been added for WebSphere Application Server v 6.0.2 and later:
com.ibm.wsspi.wssecurity.config.CertStore.Provider

This property allows Web services security, when running in WebSphere Application Server on Solaris, to use the IBMCertPath provider instead of using the Sun CertPath provider. This property is the security provider that Web services security should use when using trust anchors, and when the certificate store security provider was specified in conjunction with the certificate store.

If both the com.ibm.wsspi.wssecurity.config.CertStore.Provider is specified and a security provider is specified for the certificate store, the certificate store security provider will override the setting for com.ibm.wsspi.wssecurity.config.CertStore.Provider.

Hardware cryptographic requests with card-related exceptions must use cryptographic software to complete requests successfully

Cause:

Hardware cryptographic device-related exceptions might be seen when the machine is experiencing a heavy load. The requests complete successfully because cryptographic software is used instead for the particular operation that received the exception. However, the hardware cryptographic device is not used.

The following exceptions have been seen when running stress tests:

```
CWWSS5601E: The following exception occurred while decrypting the message:
com.ibm.pkcs11.PKCS11Exception: Another operation is already active
```

and

```
CWWSS5601E: The following exception occurred while decrypting the message: Key handle is invalid:
com.ibm.pkcs11.PKCS11Exception: Key handle is invalid
```

This problem only occurs when the hardware cryptographic card is handling a large number of operations.

Solution:

There are no known workarounds. However, the requests complete successfully because the software cryptography is used when the hardware cryptography fails for an operation.

Trace and logging for WSIF

The Web Services Invocation Framework (WSIF) offers trace points at the opening and closing of ports, the invocation of services, and the responses from services. WSIF also includes a `SimpleLog` utility that can run trace when you are using WSIF outside of WebSphere Application Server.

About this task

If you want to enable trace for the WSIF API within WebSphere Application Server, and have trace, stdout and stderr for the application server written to a well-known location, see Enabling tracing and logging.

To trace the WSIF API, you need to specify the following trace string:

```
wsif=all=enabled
```

To enable the WSIF `SimpleLog` utility, through which you can run trace when using WSIF outside of WebSphere Application Server, complete the following steps:

1. Create a file named `commons-logging.properties` with the following contents:

```
org.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryImpl
org.apache.commons.logging.Log=org.apache.commons.logging.impl.SimpleLog
```

2. Create a file named `simplelog.properties` with the following contents:

```
org.apache.commons.logging.simplelog.defaultlog=trace
org.apache.commons.logging.simplelog.showShortLogname=true
org.apache.commons.logging.simplelog.showdatetime=true
```

3. Put both these files, and the `commons-logging.jar` file, on the class path.

Results

The `SimpleLog` utility writes trace to the `System.err` file.

UDDI registry troubleshooting

This topic describes actions that you can take, such as enabling UDDI trace, to resolve errors encountered when using the UDDI registry.

About this task

When the UDDI registry is running, it might issue messages to report events or errors. Use these messages as your first aid to problem determination. For further assistance, review the list of common errors that you might encounter when setting up or using the UDDI registry. If you need more details about the cause of a problem, turn on tracing for UDDI.

1. Refer to “UDDI registry common errors” to perform initial troubleshooting.
2. Use tracing to obtain more information about the problem. To enable or disable trace using the administrative console, refer to Enabling tracing and logging; the component for the UDDI registry is `com.ibm.uddi`. You can enable tracing of the UDDI registry at several levels of granularity. For example, to enable all UDDI registry tracing, specify

```
com.ibm.uddi.*=all
```

3. Use other knowledge bases as described in Searching knowledge bases.
4. If you cannot find the cause of a problem, refer to “UDDI registry problem reporting” on page 33.

UDDI registry common errors

This topic describes some of the common causes of errors that you might encounter while using the UDDI registry, and their suggested solutions.

- The first start of the UDDI application may take some time to complete:
  - When you start the UDDI registry application for the first time with a new UDDI registry database, it must perform UDDI initialization, which occurs automatically for a default UDDI node, or when requested for a customized UDDI node. UDDI initialization populates the UDDI registry with pre-defined data and entities, and can therefore take some time to complete. This is expected behavior. Note that this only occurs on the first start, not subsequent starts, of the UDDI application.
  - If you use wsadmin to issue a command to start the UDDI application, then depending on your TCP timeout settings, this request might time out while waiting for UDDI to complete initialization. The UDDI initialization and the starting of the UDDI application will continue unaffected by this timeout, and will complete normally.
- There is a limitation concerning URL rewriting causing JavaScript™ syntax errors on several Web pages in the UDDI User Console. Because of this, cookies must be enabled in client browsers, the application server must have cookies enabled as the session tracking mechanism, and URL rewriting must be disabled.
- If you are attempting to use a remote DB2® database and are experiencing problems attaching to the remote system, one of the possible causes might be IP addressing. You should not have this problem if the remote system is using a static IP address, however if the remote system is using DHCP, the two systems must be aware of each others subnet mask.
- If you cannot see your UDDI node in the administrative console list of available nodes, check that the UDDI application is started on the relevant node/server.
- If you are unable to issue UDDI requests; for example, you start the UDDI user console and get errors when trying to publish or inquire, it is possible that:
  1. the database is not currently loaded or configured. Check the output from creating the database.
  2. the database is not correctly configured. Check the JDBC provider and datasource definitions are correct. This can be validated by using the Test Connection button on the administrative console.
  3. the UDDI node is not initialized. Check the UDDI node page on the administrative console. If the entry for the node in question does not show as activated, or deactivated, try to initialize the node having set any policies or properties first.
  4. the UDDI node is currently deactivated, while UDDI runtime settings are being updated. Check the UDDI node page on the administrative console. If the entry for the node in question shows as deactivated, then wait for it to change to activated, and then retry the request.

UDDI registry problem reporting

This topic describes information, such as log files, that you should supply to IBM if you have performed problem determination on the UDDI registry component, but cannot resolve the problem.

Before reporting a problem, perform problem determination as described in “UDDI registry troubleshooting” on page 32. If you report a problem with the UDDI registry component to IBM, supply the following information:

- A detailed description of the problem. For example, if you were using a UDDI client, describe the client error seen and the type of client program, and provide details of the request issued. If you were using the UDDI user interface, describe the exact sequence you followed, and the result.
- The build date and time of the version you are using. This can be obtained as follows:
  - In the *profile_root*/installedApps/*cell_name* subdirectory of the WebSphere installation location, you will find a subdirectory called UDDI_Registry.*node_name*.*server_name*.ear, where *node_name* is the name of the node into which the UDDI registry application is installed, and *server_name* is the name of the server. Within that subdirectory, you will find a file called version.txt. Include the contents of this file as part of your information.
  - If the UDDI registry has been started with tracing enabled for the UDDI component, you should find a trace entry in the WebSphere trace log that includes the strings ″UDDIMessageLogger″ and ″UDDI Build :″ followed by the build date and time, and the build system. Also include this information.
- Other environment information, such as:
  - Platform.
  - Database product.
  - Whether the database is local or remote, or for Apache Derby, whether it is embedded or networked.
  - Whether the server is in a standalone or a network deployment configuration.
  - If the server is in a network deployment configuration, whether it is in a cluster or not.
  - Whether the UDDI node is default or customized.
  - The language used, for example, Chinese.
- Any relevant log files and trace files.
  - If the problem occurred while setting up and installing the UDDI registry application using the setup script uddiDeploy.jacl, supply the log output from running the script. (If you did not redirect the output from the script file to a log file, rerun the script, this time redirecting the output as described in the sections Setting up a customized UDDI node or Setting up a default UDDI node.) The log file is written to the directory from which you ran the setup script. For detailed information on setting up UDDI nodes, see the chapter Setting up and deploying a new UDDI registry in the *Developing and deploying applications* PDF book.
  - If the problem occurred while removing the UDDI registry application using the remove script, uddiRemove.jacl, supply the log output from running the script. The log file is written to the directory from which you ran the remove script. For mode information about this task, see Removing a UDDI registry node in the *Administering applications and their environment* PDF book.
  - If the problem occurred while running the UDDI registry, enable UDDI tracing (if not already enabled) and supply the trace log from the logs directory of the application server on which the UDDI registry was running. See “UDDI registry troubleshooting” on page 32 for details on how to enable UDDI tracing. The Trace string that is most useful for initial analysis of problems by IBM is ″com.ibm.uddi.*=all″.
  - Also supply the WebSphere log files system.out and system.err.
  - Supply details of the version of IBM WebSphere Application Server you are running by running the command `versioninfo` on the application server node and directing the output to a log file.
- If appropriate, any application code that you are using and the output produced by the application code.
- UDDI utilizes First Failure Data Capture (FFDC) to capture data for unexpected UDDI errors.

  Run the WebSphere collector tool to collect the FFDC data and send the resulting jar to IBM. See Running the collector tool.

Chapter 3. Service integration

Resolving indoubt transactions

Use this task to resolve indoubt transactions and the messages associated with them.

About this task

Transactions might become stuck in the indoubt state indefinitely because of an exceptional circumstance such as the removal of a node causing messaging engines to be destroyed. When a transaction becomes indoubt, it must be committed or rolled back so that normal processing by the affected messaging engine can continue.

You can use the administrative console to display the messages causing the problem (see Listing messages on a message point). If there are messages involved in an indoubt transaction, the identity of the transaction is shown in a panel associated with the message. You can then resolve the transaction in two ways:

1. Using the server's transaction management panels
2. Using methods on the messaging engine's MBean

You must first attempt to resolve the indoubt transaction using the application server transaction management MBean interfaces. These are documented in Managing active and prepared transactions using scripting. Use the scripts for all application servers which may have been coordinating transactions, including Messaging actions, for the default messaging provider. If the transaction identity is known by the transaction manager scripts, use those scripts to resolve the transactions. This will consistently resolve all resources (including Messaging) within a global transaction.

If the transaction identity is not known to the transaction manager scripts that run on any application server, or if the application server hosting the transaction manager cannot be recovered, it is possible to use methods on the SIBMessagingEngine MBean to resolve the Messaging part of a transaction independently from the global transaction. The choice to commit or rollback the transaction must be made manually.

The following methods on the messaging engine's MBean can be used to get a list of transaction identities (xid) and to commit and roll back transactions:

- getPreparedTransactions()
- commitPreparedTransaction(String xid)
- rollbackPreparedTransaction(String xid)

To invoke the methods, you can use a wsadmin command, for example, you can use a command of the following form to obtain a list of the indoubt transaction identities from a messaging engine's MBean:

```
wsadmin>AdminControl.invoke(AdminControl.queryNames("type=SIBMessagingEngine,*").
splitlines()[0] , "getPreparedTransactions")
```

Note: The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Alternatively, you can use a script such as the following to invoke the methods on the MBean:

```
import sys

mebeans=AdminControl.queryNames("type=SIBMessagingEngine,*").splitlines()

for mebean in mebeans:
  input=0
  meName=""
  print "--- Start ME: ---------------"
  print mebean
  print "-----------------------------"
  while input>=0:
    xidList=AdminControl.invoke(mebean , "getPreparedTransactions").splitlines()
    print "--- Prepared Transactions ---"
    index=0
    for xid in xidList:
      print "  Index=%s XID=%s" % (index , xid)
      index+=1
    print "------- End of list ---------"
    print "Select index of XID to commit/rollback"
    print "(or press enter to skip to next ME):"
    input=int(sys.stdin.readline().strip())

    if input<0:
      print "No index selected."
    else:
      xid=xidList[input]
      print "Enter c to commit or r to rollback XID %s" % xid
      input=sys.stdin.readline().strip()
      if input=="c":
        print "Committing xid=%s" % xid
        AdminControl.invoke(mebean , "commitPreparedTransaction" , xid)
      if input=="r":
        print "Rolling back xid=%s" % xid
        AdminControl.invoke(mebean , "rollbackPreparedTransaction" , xid)
    print
  print "--- End ME --------------------"
  print

print "No more ME definitions found, exiting"
```

This script lists the transaction identities of the transactions together with an index. You can then select an index and commit or roll back the transaction corresponding to that index.

1. Use the administrative console to find the transaction identity of messages that have indoubt transactions.
2. Optional: If a transaction identity appears in the transaction management panel, commit or roll back the transactions as required.
3. Optional: If a transaction identity does not appear in the transaction management panel, use the methods on the messaging engine's MBean. For example, use a script to display a list of transaction identities for indoubt transactions. For each transaction:
   a. Enter the index of the transaction identity of the transaction.
   b. Optional: Enter `c` to commit the transaction.
   c. Optional: Enter `r` to roll back the transaction.
4. To check that transactions are no longer indoubt, restart the server and use the transaction management panel, or methods on the messaging engine's MBean to check.

Restoring a data store and recovering its messaging engine

When a failure occurs which cannot be dealt with by the system, you can restore the data store or data stores from a backup. Use this task to restore a backup of a data store and to recover its associated messaging engine afterward.

About this task

You should also restore the configuration files for the system, to ensure that it functions as it did at the time the backup was taken, for more information about why you should do this see The service integration environment backup. After you have restored the data store, you must restart the associated messaging engine.

When you restart a messaging engine after restoring a backup you must start it in **Restart after restore** mode, to minimize the effects of the messaging engine not being synchronized with any other messaging engines it was in communication with before the failure. If you restart the messaging engine in **Normal** mode, some of the new messages produced at this messaging engine might be discarded by the receiving messaging engine, for an indeterminate amount of time after restart. In **Restart after restore** mode, previously transmitted messages might be resent, potentially creating duplicates of messages that were produced before the backup was taken. However new messages are not lost or duplicated (if this is specified by the quality of service for the message).

You can restart a messaging engine in **Restart after restore** mode only by using the wsadmin client; you cannot do it from the administrative console. You must only start a messaging engine in this mode when starting the messaging engine for the first time after restoring the backup. After the initial restart, you can perform further restarts as normal.

Restart after restore mode is ignored if you start the server in **Recovery** mode. If you require both a **Recovery** mode start and a **Restart after restore** mode start:

1. Start the server in recovery mode
2. Wait for the startup to complete and for the server to stop
3. Start the messaging engine in **Restart after restore** mode

If you see the following message in the JVM System output file, it might indicate that you have restored from a backup and restarted the messaging engine without using the **Restart after restore** mode.

```
CWSIP0784E: Messaging engine: receivingME received a message from
messaging engine: producingME that was not expected.
```

To resolve this issue, stop the messaging engine and restart it in **Restart after restore** mode.

Note: This message might also appear in other situations, so you should restart the messaging engine in **Restart after restore** mode only if you know you have restored a backup.

For information on the JVM System output file and how to view it, see Viewing the JVM logs.

You can recover any number of messaging engines at the same time, by following the actions below for each messaging engine in turn.

1. Change the initial state of the messaging engine to **Stop**, so that the messaging engine will not be automatically restarted by a server process:
   a. Use the administrative console to select the messaging engine by clicking **Service integration** → **Buses** → ***bus_name*** → **[Topology] Messaging engines** → ***engine_name***.
   b. In the **Initial state** list, click **Stopped**.
   c. Click **OK**.
2. Save your changes to the master configuration, ensuring that you select the **Synchronize changes with Nodes** check box.
3. Stop the messaging engine if it is running (see Stopping a messaging engine for instructions on how to do this). If the messaging engine does not respond, stop the server process that is hosting the messaging engine.
4. Restore the backup of the data store that is accessed by the messaging engine, by referring to Restoring a data store.
5. Restore the backup of the configuration files using the backupConfig command (see Backing up and restoring administrative configurations). This backup should have been taken at the same time as the data store backup.
6. Restart any servers that were stopped by the failure.
7. Restart the messaging engine in **Restart after restore** mode by performing the following steps:

a. Start the wsadmin client.

Note: The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

For more information about the wsadmin client, see Wsadmin tool.

b. Invoke the start command using the **FLUSH** parameter, on the MBean for the messaging engine, for example:

```
wsadmin>myME=AdminControl.queryNames("type=SIBMessagingEngine,*").splitlines()[0]
wsadmin>AdminControl.invoke(myME , "state")
'Stopped'
wsadmin>AdminControl.invoke(myME , "start" , ["Flush"])
wsadmin>AdminControl.invoke(myME , "state")
'Started'
```

A number of messages might be output to the JVM SystemOut.log file to indicate the progress of the restart process.

8. Check the JVM SystemOut.log file for the following message that indicates that the restart was successful, in other words, no failures occurred while attempting to restart the messaging engine.

```
CWSIP0783E: Messaging engine: messagingEngine started,
flush of all delivery streams completed.
```

If this message does not appear, a failure has occurred which has prevented the messaging engine from restarting. Resolve the cause of the failure and repeat the **Restart after restore** procedure until the restart is successful.

Messaging engine failover between Version 6.x and Version 7.0

It is not permissible to failover a messaging engine using a file store onto a WebSphere Application Server Version 6.x server. If you have a cluster as a bus member that consists of a mixture of Version 6.x and Version 7.0 servers, you must modify the high availability policy to prevent this.

About this task

To prevent failover of a Version 7.0 messaging engine to a Version 6.x server, the high availability policy for the messaging engine should be modified so that the cluster is effectively divided into sets of servers at the different versions and the messaging engine is restricted to the servers at Version 7.0.

Problem solving for messaging engine data stores

Obtain an overview of understanding problems that can occur with a data store.

About this task

- "Diagnosing problems with data store exclusive access locks"
- "Diagnosing problems with your data store configuration" on page 40
- "Avoiding failover problems when you use DB2 v8.2 with HADR as your data store" on page 40

Diagnosing problems with data store exclusive access locks

Diagnose the causes of problems with data store exclusive access locks and examine possible causes to the problems.

About this task

Each messaging engine establishes an exclusive lock on its data store. While the messaging engine is running, it maintains that lock to ensure the integrity of the data within the data store.

Compare your symptoms with those listed in the following table and examine the possible solutions:

Symptom	Cause	Solution
The messaging engine cannot start nor failover to another server. The messaging engine writes error message CWSIS1519	The messaging engine cannot connect to the database that you specified in the data source that you configured to enable the messaging engine to access its data store.	• Check that connectivity to the database is possible using the defined data source. • If you can connect to the database, another instance of the messaging engine might hold a lock on the data store. Check for other running instances of the messaging engine. • If the messaging engine is taking over from another that was running on a different server in the same cluster, the database might not have released the database locks that comprise the data store lock. Use the administration tools for your relational database management system to examine the locks on the SIBOWNER table. If the database is still holding locks after the failure of a server, ensure that you configure the liveness checking of the network connection between the application server and the database server appropriately. For example, examine the TCP keep alive parameter.
After a delay of many minutes, the messaging engine cannot failover to another server and writes error message CWSIS1519.	Another instance of the same messaging engine is holding the data store lock, or the database has not released a data store lock that was held by a failed instance of the same messaging engine.	• If you can connect to the database, another instance of the messaging engine might hold a lock on the data store. Check for other running instances of the same messaging engine. Only one instance of each messaging engine can run in a cluster at a given time. • After the failure of an instance of a messaging engine or an application server that is running in a cluster, the database might not have released the database locks that comprise the data store lock. Use the administration tools for your relational database management system to examine the locks on the SIBOWNER table. If the database is still holding locks after the failure of a server, ensure that you configure the liveness checking of the network connection between the application server and the database server to enable prompt failover. For example, examine the TCP keep alive parameter. You might need to use your database administration tools to force the release of the lock on the data store.
The messaging engine fails to start and writes error messages CWSIS1535 and CWSIS1519	The identifiers in the SIBOWNER table do not match those of the messaging engine.	• Check that the data source that you configured for the messaging engine refers to the correct database. • If the **MEUUID** identifiers do not match, check that a previous messaging engine did not use the same tables. If the tables already exist, DROP the tables and CREATE them again for the new messaging engine. • If the **INCUUID** identifiers do not match, another instance of the same messaging engine is running and has acquired the lock. Check for other running instances of the messaging engine.

Symptom	Cause	Solution
The messaging engine starts but then stops and writes error message CWSIS1519.	The messaging engine has lost its lock on the data store.	• Check that you have connectivity to the database through the data source that you specified. The messaging engine might have lost network connectivity and be unable to maintain a connection with the database. • If you can connect to the database, another instance of the messaging engine might have started and obtained a lock on the data store. Check for other running instances of the messaging engine.

Diagnosing problems with your data store configuration

Find out how to diagnose problems that are caused by your data store configuration and possible solutions to these problems.

About this task

The following problems depend on the database that you use with your data store configuration and the level of that database:

- Examine this section if your messaging engine uses a Sybase database for its data store. When you create your Sybase server:
  - Ensure that you create the database server with a page size of at least 4k.
  - Ensure that you set the **lock scheme** property on your server to the value *datarows*. This avoids the possibility of a deadlock on the data store tables.
- Examine this section if your messaging engine uses an Informix® database for its data store and the messaging engine is unable to access its data store. When you configure your messaging engine to use an Informix database, you must specify the schema name in lowercase letters. For a full description of the configuring procedure, refer to Configuring a messaging engine data store to use a data source.

Avoiding failover problems when you use DB2 v8.2 with HADR as your data store

Use this task to avoid problems that can occur when a messaging engine that is configured to use DB2 v8.2 with the High Availability Data Recovery (HADR) feature for its data store terminates if the DB2 database fails over.

About this task

If you use the High Availability Data Recovery (HADR) feature of DB2, note the following restrictions:

- The messaging engine default messaging provider supports only the synchronous and near-synchronoous synchronization modes of HADR. The default messaging provider does not support asynchronous HADR configurations.
- The TAKEOVER BY FORCE command is permitted only when the standby database is in peer state, or when the standby database had last changed from peer state to its current state (such as disconnected state).

Problem solving for messaging engine file stores

Obtain an overview of improving the performance of messaging engine file stores and understanding problems that can occur with file stores.

Diagnosing problems with accessing file store files

Diagnose the causes of problems with accessing files in the file store and examine possible causes to the problems.

About this task

Compare your symptoms with those listed in the following table and examine the possible solutions:

Symptom	Cause	Solution
The messaging engine fails to start and writes error messages CWSIS1579E and CWSIS1583E.	The log file for the file store is locked by another process and cannot therefore be opened by this messaging engine. The messaging engine has attempted to retry to connect to the log file but has reached its retry limit and stopped.	• Check that no other instance of this or any other messaging engine is running using the same set of file store files.
The messaging engine fails to start and writes error messages CWSIS1580E and CWSIS1583E.	The file store was unable to find or create its log file in the location specified in the messaging engine configuration. The messaging engine has attempted to retry to connect to the log file but has reached its retry limit and stopped.	• Check that the directory specified for the file store log file exists and is accessible by the username that the WAS process is running under. • If the file store files are accessed via a network, check that a connection between the messaging engine server and the directory on the file server is available.
The messaging engine stops unexpectedly and writes message CWSIS1590E.	The file store has encountered an unexpected problem and has had to stop to ensure that it has a reliable copy of its data on disk. If configured to do so, the messaging engine will failover to a new instance and attempt to re-start.	• Check your server logs for any error messages output before message CWSIS1590E to see if they explain the shutdown. • If the file store files are accessed via a network, check that a connection between the messaging engine server and the file server is available.

Reducing file store file sizes

It is possible to reduce the sizes of the log file and store files in the configuration. However, such reduction is only possible when certain conditions are met.

About this task

Reducing the file sizes can be difficult in practice due to the following reasons:

1. It is not possible to shrink the files down below the amount of space that their contents currently consume.
2. There is no support for compaction of the contents of the permanent store file and temporary store file so fragmentation may keep these store files artificially large. As such, when the values of the file sizes in the configuration are reduced and the messaging engine restarted, it might not be possible to change the sizes of the files to the desired values.

When this situation occurs, the messaging engine emits warning messages to `SystemOut.log` and continues to use the existing values. It attempts to apply the configuration changes repeatedly each time it starts until it succeeds.

Problems might also arise when the file store file sizes are set too small.

Listing messages on a message point

Use this task to list the messages that exist on a message point for a selected bus destination or messaging engine.

About this task

To display a list of messages on a message point, use the administrative console to complete the following steps:

1. In the navigation pane, click **Service integration** → **Buses**.
2. In the content pane, click the name of the service integration bus.
3. Optional: To list the message points for a bus destination, complete the following steps:
   a. In the content pane, under **Destination resources**, click **Destinations**.
   b. Click the destination name.
4. Optional: To list the message points for a messaging engine, complete the following steps:
   a. In the content pane, under **Topology**, click **Messaging engines**
   b. Click the messaging engine name.
5. Under Additional Properties, click **Message points** This displays a list of message points in the content pane.
6. Click the message point name. This displays the properties of the destination localization in the content pane.
7. Click the Runtime tab.
8. Under Additional Properties, click **Messages**

Results

A list of messages on the selected message point is displayed in the content pane.

What to do next

You can select one or more messages to act on; for example, to display the message content, delete messages.

Deleting messages on a message point

Use this task to delete one or messages that exist on a message point for a selected bus destination or messaging engine.

About this task

You should not normally need to delete messages on a message point. This task is intended as part of a troubleshooting procedure.

To delete one or messages on a message point, use the administrative console to complete the following steps:

1. List the messages on the message point.
2. In the content pane, click the check box next to each message you want to delete. Alternatively, you can select all messages in the list by clicking the Select all items button ☑
3. Click **Delete**

Results

The selected messages are removed from the list.

Troubleshooting service integration message problems

This topic contains links to tasks that will help you to investigate problems with messages, such as messages not arriving or being consumed, or poison messages.

About this task

If you are having problems with messages not behaving as you expect, use the links below to navigate to the topic that is appropriate for your problem.

- "Understanding why best effort messages are being discarded"
- "Investigating why a queue is full"
- "Investigating why a topic space is full" on page 45
- "Investigating why point-to-point messages are not arriving" on page 47
- "Investigating why point-to-point messages are not being consumed" on page 52
- "Investigating why publish/subscribe messages are not arriving at a subscription" on page 59

Understanding why best effort messages are being discarded

A reliability level of *best effort* means that messages might be lost during normal functioning of the system, for example if the connection used to send the messages is busy. Although this is normal and expected, you might want to investigate the reasons for messages being lost.

About this task

Use this task when best effort messages are being discarded by a running system, and you want to understand the possible causes. For more information about "best effort" and other message reliability levels, see Message reliability levels.

The following list explains some of the reasons for losing best effort messages:

- The destination queue or topic space is already full to a level higher than the high message threshold. To check whether this is the case, click **Service integration -> Buses -> *bus_name* ->** [Additional Properties] **Destinations -> *destination_name*** and under **Message points** click the relevant point type (for example, **Queue points**). Click the relevant message point to display its general properties, and compare the values of the **High message threshold** and **Current message depth** fields.
- The connection to the target system is down.
- The connection to the target system is busy. Any best effort messages that cannot be sent will be discarded.
- The system in general is busy, for example a messaging engine might be occupied processing higher reliability messages for another destination.
- There is a temporary network problem. Look in the error log for more information.

Investigating why a queue is full

This topic describes how to investigate why a queue on a service integration bus is full.

About this task

When a queue becomes full, exceptions are returned when you attempt to produce a message to that queue. The most probable reason for a queue filling up is that the producing application is producing messages faster than they can be consumed by the consuming application, although causes can also include broken communication links or errors in the consuming application.

1. Click **Service integration** → **Buses** → ***bus_name*** → **[Destination resources] Destinations**, then click the name of the queue that is full.
2. Click **[Related Items] Application resources topology**, then use the Application resources for this destination panel to inspect the configuration of the applications and JMS resources that are using the destination.

   This panel can help you find the cause of the problem by giving you a high level view of many relevant resources.
3. Click **Service integration** → **Buses** → ***bus_name*** → **[Destination resources] Destinations** → ***queue_name*** → **[Message points] Queue points** → ***queue_point_name***, then on the **Runtime** tab review the value of the **Current message depth**. If this value increases steadily, the producing application is outpacing the consumer.

   Note: If the destination has multiple queue points, or is mediated, perform the following checks for each message point the message could have been sent to or consumed from.
4. Determine which messaging engines the producing and consuming applications are connected to.
5. If the producing and consuming applications are connected to different messaging engines, the messages are being routed via a remote queue point. On the producer's messaging engine, click **Remote queue points** and then click the queue point that represents the consumer's queue point. Review the number of current outbound messages. If the number of current messages is low, the problem does not lie with the remote queue point; check that the consuming application is started and is consuming messages without error. If the number of current messages is approaching the high message threshold, perform the following checks:
   - Check that the two messaging engines can communicate with each other, see "Service integration troubleshooting: Checking the communication between two messaging engines in a bus" on page 45. If the messaging engines can communicate, reduce the rate at which messages are produced. If the messaging engines cannot communicate, resolve the failure. If you encounter problems processing the backlog of messages once communication is restored, and the backlog does not contain any messages that are vital, consider deleting all the messages on the remote message point. To delete the messages, select the relevant remote message point and click **Delete all messages**.

     Note: You will not be able to recover the messages once they have been deleted.
   - Check that messages are not being trapped in the Committing state. If they are, a resource manager, such as a database, has hung. Resolve the issue with the resource manager. If this fails, note the **Transaction ID** of the message and click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **Runtime** → **[Additional Properties] Transaction Service** to display the general properties for the transaction service, including numbers of transactions. Use the **Review** links to resolve the transaction whose **Global ID** matches the transaction ID of the message.

Determining which messaging engine an application is connected to

If your application fails to receive or produce a message, you might want to find out which messaging engine it is connected to, as part of troubleshooting the problem.

1. If your application is a JMS application, examine its connection factory as the messaging engine name might be specified there.
2. If your application is not a JMS application, or its connection factory does not specify the messaging engine name, use one of the following methods to determine which messaging engine the application is connected to:
   - Within the application code, after the application has obtained a valid Connection object, add a call to the toString() method of that object. The connected messaging engine name will be clearly listed when you rerun the application.

- Enable the SIBJms_External trace component and rerun the application. Inspect the generated trace for a reference to the connected messaging engine name.

Results

Be aware that the messaging engine name returned by either of these methods relates to the rerun of the application. It is possible that the original failing instance of the application was connected to a different messaging engine in the bus.

Service integration troubleshooting: Checking the communication between two messaging engines in a bus

If you are troubleshooting a problem with your service integration system, you might want to check that two messaging engines can communicate with each other.

1. Check that both messaging engines are running.
2. For each messaging engine:
   a. Check the system log for a CWSIT0028I message that relates to the two messaging engines in question. This message indicates that the two messaging engines are successfully communicating with each other.
   b. Find the most recent instance (there may be only one) of the CWSIT0028I message for the two messaging engines, and check the log to make sure that a CWSIT0029I message for these two messaging engines does not appear later in the log. This message indicates that the communication connection between the two messaging engines has failed.

   If either of these checks fails, inspect the log for indications of the cause of the failure, and follow the suggested actions to rectify the problem.

Investigating why a topic space is full

This topic describes how to investigate why a topic space on a service integration bus is full.

About this task

When a topic space becomes full, exceptions will be returned when you attempt to publish a message to that topic space. The most probable reason for a topic space filling up is that the publishing application is producing messages faster than they can be consumed by the subscribing application or applications. However there could be other causes, such as dormant subscribers or broken communications links.

Another possible cause is a regular increase in message traffic, for example at certain times of day. Consider increasing the high message threshold to overcome this problem.

1. Click **Service integration** → **Buses** → ***bus_name*** → **[Destination resources] Destinations** to display a list which includes all the topic spaces on that bus. Click the name of the topic space that is full.
2. Click **[Message Points] Publication points**.
3. Click the name of a publication point, then on the **Runtime** tab review the value of the **Current message depth**. If this value increases steadily, the publishing application is outpacing the subscribers. Click **Subscriptions** to display the subscriptions for the topic space. For each subscription, click on the subscription name and examine the **Current message depth**. If all the subscriptions are filling up, reduce the rate at which the publishing application is publishing messages.

   Note: If the topic space is mediated, perform the following checks for each mediation point the message could have been sent to or consumed from.
4. If only one subscription is filling up, the problem lies with the related subscribing application. If the subscription is nondurable, modify the subscribing application to increase the speed of consumption.
5. If the subscription is a durable subscription, click **Messages** and ensure that the message at the top of the list changes with time; this indicates that the subscribing application is actually consuming

messages. If the message does not change but the application is running, either delete the subscription or increase the high message threshold of the publication point.

6. Determine which messaging engines the publishing and subscribing applications are connected to, see “Determining which messaging engine an application is connected to” on page 44.
7. If the publishing and subscribing applications are connected to different messaging engines, the messages are being routed via a remote queue point. On the publisher’s messaging engine, click **Remote publication points** and then click the publication point that represents the subscriber’s publication point. Review the number of current outbound messages. If the number of current messages is low, the problem does not lie with the remote message point. If the number of current messages is approaching the high message threshold, perform the following checks:
   - Check that the two messaging engines can communicate with each other, see “Service integration troubleshooting: Checking the communication between two messaging engines in a bus” on page 45. If the messaging engines can communicate, reduce the rate at which messages are published. If the messaging engines cannot communicate, resolve the failure. If you encounter problems processing the backlog of messages once communication is restored, and the backlog does not contain any messages that are vital, consider deleting all the messages on the remote message point. To delete the messages, select the relevant remote message point and click **Delete all messages**.

     Note: You will not be able to recover the messages once they have been deleted.
     To avoid the messages building up again, click **Topics**, then click **Clear all**. No more messages will be sent to this remote publication point. To reset the topic list, restart the messaging engine.
   - Check that messages are not being trapped in the Committing state. If they are, a resource manager, such as a database, has hung. Resolve the issue with the resource manager. If this fails, note the **Transaction ID** of the message and click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **Runtime** → **[Additional Properties] Transaction Service** to display the general properties for the transaction service, including numbers of transactions. Use the **Review** links to resolve the transaction whose **Global ID** matches the transaction ID of the message.

Determining which messaging engine an application is connected to

If your application fails to receive or produce a message, you might want to find out which messaging engine it is connected to, as part of troubleshooting the problem.

1. If your application is a JMS application, examine its connection factory as the messaging engine name might be specified there.
2. If your application is not a JMS application, or its connection factory does not specify the messaging engine name, use one of the following methods to determine which messaging engine the application is connected to:
   - Within the application code, after the application has obtained a valid Connection object, add a call to the toString() method of that object. The connected messaging engine name will be clearly listed when you rerun the application.
   - Enable the SIBJms_External trace component and rerun the application. Inspect the generated trace for a reference to the connected messaging engine name.

Results

Be aware that the messaging engine name returned by either of these methods relates to the rerun of the application. It is possible that the original failing instance of the application was connected to a different messaging engine in the bus.

Service integration troubleshooting: Checking the communication between two messaging engines in a bus

If you are troubleshooting a problem with your service integration system, you might want to check that two messaging engines can communicate with each other.

1. Check that both messaging engines are running.
2. For each messaging engine:
   a. Check the system log for a CWSIT0028I message that relates to the two messaging engines in question. This message indicates that the two messaging engines are successfully communicating with each other.
   b. Find the most recent instance (there may be only one) of the CWSIT0028I message for the two messaging engines, and check the log to make sure that a CWSIT0029I message for these two messaging engines does not appear later in the log. This message indicates that the communication connection between the two messaging engines has failed.

   If either of these checks fails, inspect the log for indications of the cause of the failure, and follow the suggested actions to rectify the problem.

Investigating why point-to-point messages are not arriving

This topic describes how to investigate why point-to-point messages are not arriving at a destination on a service integration bus.

About this task

Use this topic if you have an application that is producing point-to-point messages in a service integration system, and the messages are not arriving at their destination.

Perform the following preliminary checks before starting the investigation:

- Check that the producing application is producing messages correctly:
  - Check that there are no failures in the application.
  - Check that the name of the destination is correct.
  - Check that the transaction used to produce the message was committed without any exceptions.
  - Check that the application is allowing sufficient time for messages to be delivered; messages are transmitted asynchronously between messaging engines, so if the messages are being routed through a remote message point there may be a slight delay before they are delivered. The length of the delay is dependent upon factors such as system capacity and loading.
- Check the producing application to see if it is giving the messages a short expiry time. If this is the case, the messages may be expiring before they arrive, or before they can be processed by the receiving messaging engine.
- Examine the relevant exception destination to see if the messages appear there. If they do, use the information contained within the messages to understand why they have arrived at the exception destination, and write an application (or mediation) to process the messages.
- Check the reliability of the messages. If the reliability is set to best effort, the messages can be discarded by the system during normal operation. See “Understanding why best effort messages are being discarded” on page 43 for a list of possible causes.
- Check the priority of the messages. If the priority is low, higher priority work may be delaying the messages.
- Check the system environment, for example, a busy CPU might cause delayed messages.
- Examine the error logs for exceptions.

1. Click **Service integration -> Buses ->** ***bus_name*** **->** [Destination resources] **Destinations** to display the destinations on the relevant bus. Click on the destination and check that the **Send allowed** check box is selected.
2. Stop the consuming application. Clear the **Receive allowed** check box for the destination and save the changes to the master repository. If you do not have dynamic configuration enabled, restart the messaging engine for the changes to take effect. This will prevent any consumers from consuming the test message that you will use to investigate the problem.

3. Run the producing application to produce a test message with a reliability level greater than best effort (best effort messages can be discarded during normal operation so are not useful for investigating this problem). The following steps describe how to investigate what happens to the test message.
4. Determine which messaging engine hosts the queue point for the destination to which the messages are being sent, see "Determining the location of message points for a destination on a service integration bus."
5. Click **Service integration -> Buses ->** ***bus_name*** **->** [Topology] **Messaging engines** and check that the messaging engine is running.
6. From the messaging engine panel click [Message points] **Queue points ->*****queue_point_name*** [Runtime tab] **Messages** to view the messages on the queue point. If the message is displayed, it arrived successfully at the messaging engine and the problem has cleared.
7. Determine which messaging engine the producing application is connected to, see "Determining which messaging engine an application is connected to" on page 44.
8. If the producing application is connected to a messaging engine other than the messaging engine hosting the queue point, the messages are being routed through a remote message point. Refer to "Investigating why point-to-point messages are not arriving through a remote message point" on page 49 to investigate this scenario.

Determining the location of message points for a destination on a service integration bus

When you are investigating a problem, you might need to find where the message points for a destination are located.

About this task

You may need to perform this task as part of problem determination, to find out on which messaging engine a message point is located.

Note: If you have an alias destination, the alias is resolved to the physical destination immediately after a message is produced. You can use this task to find the physical destination.

Click **Service integration -> Buses ->** ***bus_name*** **->** [Additional Properties] **Destinations** to display the destinations on the relevant bus. Review the **Type** of the destination:

- If the destination is a queue, the queue point name has the form *destination@messaging_engine_name*.
  - If the queue is localized to a cluster, there is one queue point for every messaging engine in the cluster.
  - If the queue is localized to a cluster, there is one queue point for every messaging engine in the cluster.

  A mediated queue has at least one mediation point. If the queue is localized to a cluster, there is one mediation point for every messaging engine in the cluster.
-
- If the destination is a topic space, it will have a publication point localized to every messaging engine in the bus. Messages produced to a topic space are always delivered directly to the publication point situated on the same messaging engine that the producing application is connected to. A topic space may be mediated in the same way as a queue, in which case messages will initially be directed to the one or more mediation points and then to a publication point co-located with the mediation point.

What to do next

If the destination is a queue with multiple queue points, or it is mediated using multiple mediation points, perform the problem determination for each message point that the message could have been sent to or consumed from.

Investigating why point-to-point messages are not arriving through a remote message point

This topic describes how to investigate why point-to-point messages are not arriving at a destination on a service integration bus, when the messages are being routed through a remote message point.

Before you begin

Follow the steps in “Investigating why point-to-point messages are not arriving” on page 47, which contains preliminary checks and investigative tasks that you should carry out before proceeding with this task.

About this task

You should perform this task as part of “Investigating why point-to-point messages are not arriving” on page 47. This task explains how to investigate the flow of messages in a point-to-point messaging scenario where the messages are being routed through a remote message point. The following diagram illustrates the situation. ME1 is the messaging engine that the producing application is attached to, and ME2 is the messaging engine that is hosting the queue point. These messaging engines are referred to in the following steps.

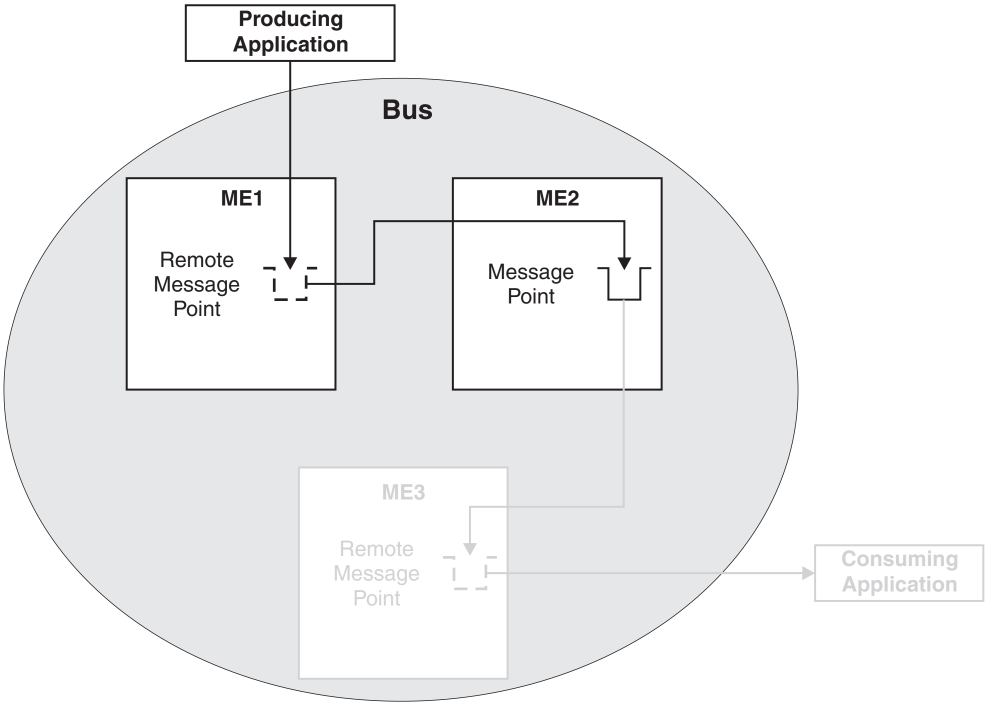

Figure 1. Point-to-point message production using a remote message point

1. Display the properties for ME1 by clicking **Service integration** → **Buses** → ***bus_name*** → **[Topology] Messaging engines** → ***engine_name***.
2. On the **Runtime** tab for ME1, click **[Remote message points] Remote queue points**, then click the remote queue point that represents the queue point on ME2. Review the value of the **Current outbound messages** field.
3. If the number of current outbound messages is greater than zero, messages have been produced but they might not have been received by ME2.

a. Check that the two messaging engines can communicate with each other, see “Service integration troubleshooting: Checking the communication between two messaging engines in a bus” on page 45.

b. Look for previous messages on the queue. If there are previous messages, and some or all of them are for ME2, wait a few moments and then refresh the view.

   - If some of the messages have disappeared from the queue, the system is currently delivering messages but is backlogged. Wait until the backlog has been cleared, then inspect the queue point on ME2 to see if the test message has arrived.
   - If none of the messages have disappeared from the queue, the transmission of messages might be blocked by a message that is trapped in the Committing state. Later messages must wait for this message to be delivered, otherwise the ordering of messages will be broken.

     If a message is trapped in the Committing state, that message is contained in an unresolved transaction. A resource manager, such as a database, might have hung. Resolve the issue with the resource manager. If this fails, note the **Transaction ID** of the message and click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **Runtime** → **[Additional Properties] Transaction Service** to display the general properties for the transaction service. Use the **Review** links to resolve the transaction whose **Global ID** matches the transaction ID of the message.

c. Examine the state of the test message:

   - If the status of the test message is “Pending send”, the message is waiting to be sent. ME2 might not be accepting messages. Perform the following checks:
     – Check that the two messaging engines can communicate with each other, see “Service integration troubleshooting: Checking the communication between two messaging engines in a bus” on page 45.
     – Check that the queue point on ME2 is not full: display the runtime properties for the queue point and compare the **Current message depth** to the **High message threshold**. If the current message depth is equal to the high message threshold, the messaging engine will not accept new messages until the queued messages have been consumed. Either restart the consumer and wait until the backlog is cleared, or delete the messages.

       Note: You will not be able to recover the messages once they have been deleted.
     – Check that configuration changes have been propagated. Ensure that ME2 is aware of the existence of the queue point by deploying the latest configuration settings to ME2’s application server.
   - If the status of the test message is “Pending acknowledgement”, the message has been sent but ME2 has either not received the message, or not processed the message. Check that there are no messages in the Committing state ahead of the test message in the transmit queue, then wait a few moments and examine the queue point again to see if the test message has arrived.If there are messages that are trapped in the Committing state, resolve this problem by referring to the following point.
   - If the test message (or another message) is in the Committing state, the message is contained in an unresolved transaction. A resource manager, such as a database, might have hung. Resolve the issue with the resource manager. If this fails, note the **Transaction ID** of the message and click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **Runtime** → **[Additional Properties] Transaction Service** to display the general properties for the transaction service. Use the **Review** links to resolve the transaction whose **Global ID** matches the transaction ID of the message.

4. If the number of completed outbound messages is greater than zero, messages have been produced and processed by ME2, but the test message has not appeared. Rerun the producing application and ensure that the number of completed outbound messages on ME1 increases (you might see the active outbound message count increase before the completed outbound message count increases).

- If the counts do not increase, the message was not produced at ME1. Check that the producing application is actually connected to this messaging engine (see "Determining which messaging engine an application is connected to" on page 44).
- If the counts do increase, the message arrived at ME2, but was either consumed, sent to the exception destination, or expired. Check for the presence of consumers, and perform the preliminary checks again.

5. If the number of current and completed messages are both zero, check that the producing application really is producing messages to this destination, by performing the relevant preliminary checks again.

What to do next

If you are still having problems, contact your IBM customer service representative.

Determining which messaging engine an application is connected to:

If your application fails to receive or produce a message, you might want to find out which messaging engine it is connected to, as part of troubleshooting the problem.

1. If your application is a JMS application, examine its connection factory as the messaging engine name might be specified there.
2. If your application is not a JMS application, or its connection factory does not specify the messaging engine name, use one of the following methods to determine which messaging engine the application is connected to:
   - Within the application code, after the application has obtained a valid Connection object, add a call to the toString() method of that object. The connected messaging engine name will be clearly listed when you rerun the application.
   - Enable the SIBJms_External trace component and rerun the application. Inspect the generated trace for a reference to the connected messaging engine name.

Results

Be aware that the messaging engine name returned by either of these methods relates to the rerun of the application. It is possible that the original failing instance of the application was connected to a different messaging engine in the bus.

Service integration troubleshooting: Checking the communication between two messaging engines in a bus:

If you are troubleshooting a problem with your service integration system, you might want to check that two messaging engines can communicate with each other.

1. Check that both messaging engines are running.
2. For each messaging engine:
   a. Check the system log for a CWSIT0028I message that relates to the two messaging engines in question. This message indicates that the two messaging engines are successfully communicating with each other.
   b. Find the most recent instance (there may be only one) of the CWSIT0028I message for the two messaging engines, and check the log to make sure that a CWSIT0029I message for these two messaging engines does not appear later in the log. This message indicates that the communication connection between the two messaging engines has failed.

   If either of these checks fails, inspect the log for indications of the cause of the failure, and follow the suggested actions to rectify the problem.

Investigating why point-to-point messages are not being consumed

This topic describes how to investigate why point-to-point messages are not being consumed from a destination on a service integration bus.

About this task

Use this topic if you did not get a response in your application because a message you were expecting did not appear on a queue. The information in this topic applies to local and remote producers, and local and remote consumers.

Perform the following preliminary checks before starting the investigation:

- Perform the following preliminary checks before starting the investigation:
- Check that the consuming application is consuming messages correctly:
  - Check that the application is started.
  - Check that the name of the destination being consumed from is correct.
- Check the producing application to see if it is giving the messages a short expiry time. If this is the case, the messages may be expiring before they can be consumed.
- Click **Service integration** → **Buses** → ***bus_name*** → **[Destination resources] Destinations** to display the destinations on the relevant bus. Click the destination and check that the **Receive allowed** check box is selected.
- Check the reliability of the messages. If the reliability is set to best effort, the messages can be discarded by the system during normal operation. See “Understanding why best effort messages are being discarded” on page 43 for a list of possible causes.
- Examine the error logs.

1. Run the consuming application and check that messages are still not being consumed.
2. Stop the consuming application.
3. Determine which messaging engine is hosting the queue point to which messages are being produced. See “Determining the location of message points for a destination on a service integration bus” on page 48.
4. Click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **[Server messaging] Messaging engines** → ***engine_name*** → **[Message points] Queue points** → ***queue_point_identifier*** → **[Runtime tab] Messages** to view the messages on the queue point. Check that there are messages present that are in the Unlocked state.
   - If there are no messages present, then there are no messages to consume. Run the producing application to produce a test message and check the queue again. If there are still no messages present, the test message has not arrived. Use the topic “Investigating why point-to-point messages are not arriving” on page 47 to investigate the problem.
   - If there are messages present but they are not in the Unlocked state, check for other consumers that are consuming from this queue point. If there are other consumers, stop them and repeat the investigation.
5. Determine which messaging engine the consuming application is connected to. See “Determining which messaging engine an application is connected to” on page 44.
   - If the consuming application is connected to the messaging engine hosting the queue point, check the consuming application for errors, in particular check that the selector in the consuming application matches the available message.
   - If the consuming application is connected to a messaging engine other than the messaging engine hosting the queue point, the messages are being routed through a remote message point. Display the runtime properties of the messaging engine that the consuming application is connected to, then display the remote message points for that messaging engine and view the list of message requests on the relevant message point. If possible, start the consuming application and ensure that it is actively trying to consume a message (the application should be in either a “receive with wait” state

or an “asynchronous consumer registered” state), then follow the instructions in “Investigating why messages are not being consumed through a remote message point or subscription point, while the application is running.” If your application cannot remain in an actively consuming state for a significant length of time (long enough to investigate the problem), follow the steps in “Investigating why messages are not being consumed through a remote message point or subscription point, while the application is stopped” on page 56.

Determining which messaging engine an application is connected to

If your application fails to receive or produce a message, you might want to find out which messaging engine it is connected to, as part of troubleshooting the problem.

1. If your application is a JMS application, examine its connection factory as the messaging engine name might be specified there.
2. If your application is not a JMS application, or its connection factory does not specify the messaging engine name, use one of the following methods to determine which messaging engine the application is connected to:
   - Within the application code, after the application has obtained a valid Connection object, add a call to the toString() method of that object. The connected messaging engine name will be clearly listed when you rerun the application.
   - Enable the SIBJms_External trace component and rerun the application. Inspect the generated trace for a reference to the connected messaging engine name.

Results

Be aware that the messaging engine name returned by either of these methods relates to the rerun of the application. It is possible that the original failing instance of the application was connected to a different messaging engine in the bus.

Investigating why messages are not being consumed through a remote message point or subscription point, while the application is running

This topic describes how to investigate why messages are not being consumed at a destination on a service integration bus, when the messages are being routed through a remote message point and the consuming application is running.

Before you begin

Follow the steps in either “Investigating why point-to-point messages are not being consumed” on page 52 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 59, whichever best suits the problem. These topics contain preliminary checks and investigative tasks that you should carry out before proceeding with this task.

About this task

You should perform this task as part of either “Investigating why point-to-point messages are not being consumed” on page 52 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 59. This task explains how to investigate the flow of messages in a scenario where the messages are being routed through a remote message point and the consuming application is started. The following diagrams illustrate two possible scenarios. In Figure 1, ME2 is the messaging engine that hosts the message point, and receives messages from the producing application through ME1. ME3 is the messaging engine that the consuming application is attached to, and hosts a remote message point which represents the message point on ME2. In Figure 2, ME2 and ME3 host publication points that are represented by remote publication points on ME1, where the producing application is attached. Subscribing application B is connected to ME3 and receives messages indirectly from ME1, via a subscription on ME2. a remote subscription point on ME 3. These messaging engines are referred to in the following steps.

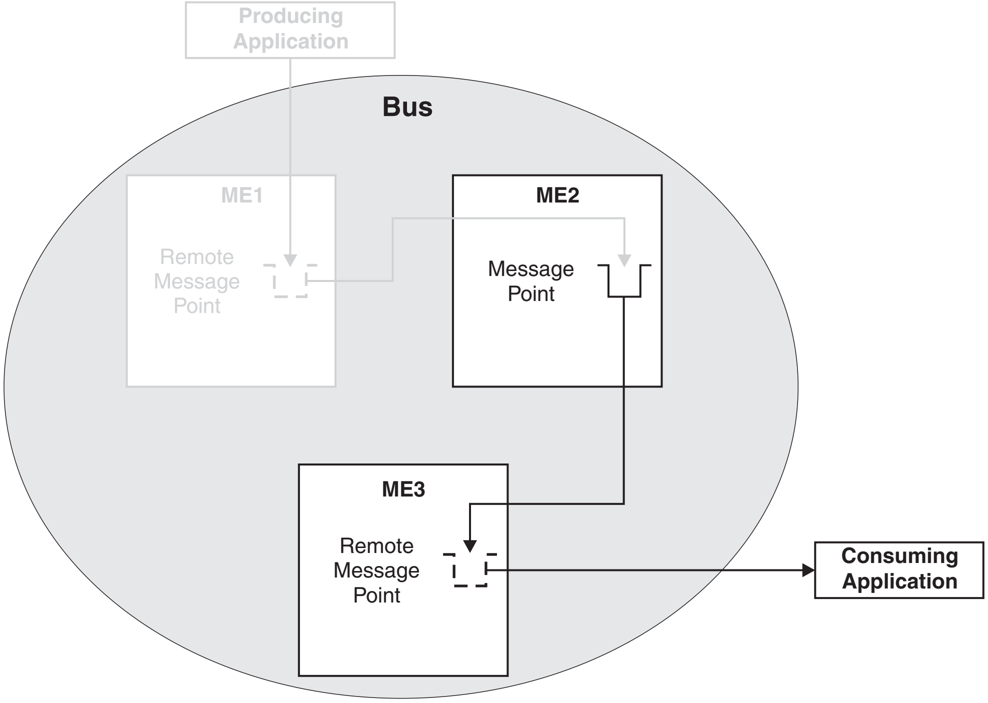

Figure 2. Point-to-point message consumption using a remote message point

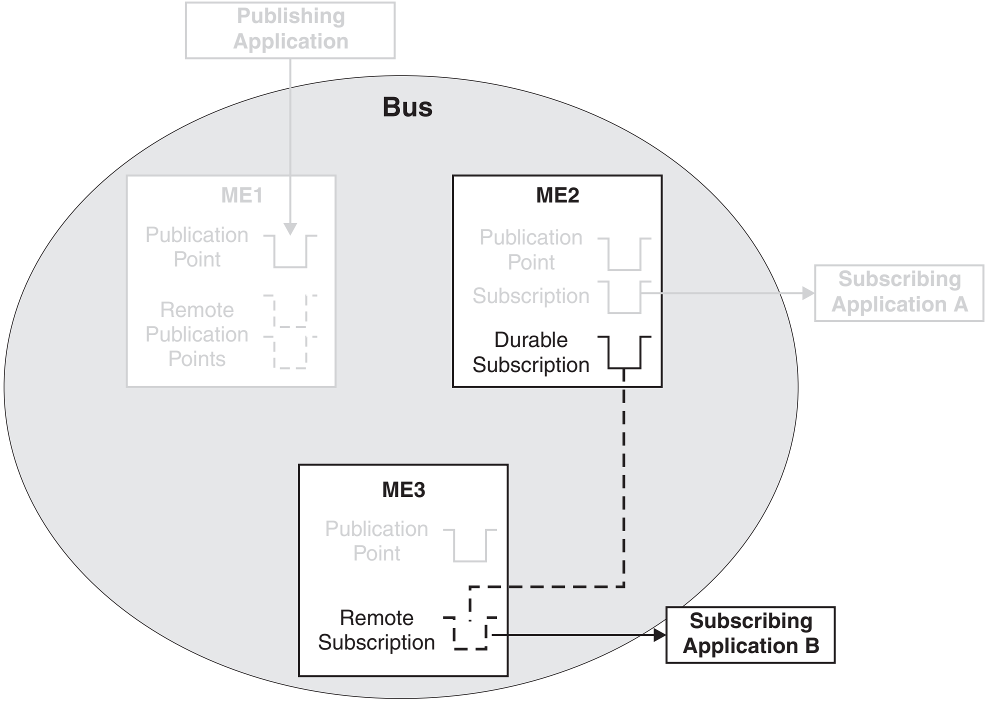

Figure 3. Publish/subscribe messaging using a remote message point

1. If you have followed the steps in “Investigating why point-to-point messages are not being consumed” on page 52 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 59 before starting this task, you should have displayed a list of message requests. Check that the list contains a request with a selector that matches an available message on the message point on ME2. If there is no such request in the list, the consuming application is not consuming; check the consuming application for errors:
   - Check that the consumer is started.
   - Check that the application is actively trying to consume:
     - If the application uses an asynchronous consumer, check that the asynchronous consumer is registered.
     - If the application is synchronous, check that the consumer is currently in a “receive with wait” state (this may require a modification to the application to extend the time that the application waits for a message).
2. Check the state of the active request:
   - If the state is Value, a message was retrieved and returned to the consuming application, but the consumption of the message has not yet completed. Check that the consuming application is correctly processing any incoming messages, for example, check that the application is committing the transaction used to consume the message.
   - If the state is Rejected, a message was retrieved and returned to the consuming application, which then rejected the message for some reason. Generally, this means that the consuming application rolled back the consume operation or an associated transaction.
   - If the state is Acknowledged, a message was returned for the request and consumed by an application. Check that the message was received by the correct application, and was not consumed by a different application.
   - If the state is Request, the message request has been sent to ME2, continue to the next check to investigate why a message has not been returned.
3. Note the **Request ID**. On ME2, display the message points for the destination, and view the message requests from ME3. Check that there is a request which matches the request ID on ME3. If there is no matching request, ME2 is not aware of the request. Check that the two messaging engines can communicate with each other, see “Service integration troubleshooting: Checking the communication between two messaging engines in a bus” on page 45.
4. Check the state of the request:
   - If the request state is Requested, the request has been received but no suitable message is available. Check that the request selector matches the available message on the message point.
   - If the request state is “Pending acknowledgement”, the request has successfully identified a matching message and attempted to transmit it to ME3. Check that the two messaging engines can communicate with each other, see “Service integration troubleshooting: Checking the communication between two messaging engines in a bus” on page 45.

What to do next

If you are still having problems, contact your IBM customer service representative.

Service integration troubleshooting: Checking the communication between two messaging engines in a bus:

If you are troubleshooting a problem with your service integration system, you might want to check that two messaging engines can communicate with each other.

1. Check that both messaging engines are running.
2. For each messaging engine:

a. Check the system log for a CWSIT0028I message that relates to the two messaging engines in question. This message indicates that the two messaging engines are successfully communicating with each other.
b. Find the most recent instance (there may be only one) of the CWSIT0028I message for the two messaging engines, and check the log to make sure that a CWSIT0029I message for these two messaging engines does not appear later in the log. This message indicates that the communication connection between the two messaging engines has failed.

If either of these checks fails, inspect the log for indications of the cause of the failure, and follow the suggested actions to rectify the problem.

Investigating why messages are not being consumed through a remote message point or subscription point, while the application is stopped

This topic describes how to investigate why messages are not being consumed at a destination on a service integration bus, when the messages are being routed through a remote message point and the consuming application is stopped.

Before you begin

Follow the steps in either “Investigating why point-to-point messages are not being consumed” on page 52 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 59, whichever best suits the problem. These topics contain preliminary checks and investigative tasks that you should carry out before proceeding with this task.

About this task

Perform this task as part of either “Investigating why point-to-point messages are not being consumed” on page 52 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 59. This task explains how to investigate the flow of messages in a scenario where the messages are being routed through a remote message point and the consuming application is stopped. The following diagrams illustrate two possible scenarios. In Figure 1, ME2 is the messaging engine that hosts the message point, and receives messages from the producing application through ME1. ME3 is the messaging engine that the consuming application is attached to, and hosts a remote message point which represents the message point on ME2. In Figure 2, ME2 and ME3 host publication points that are represented by remote publication points on ME1, where the producing application is attached. Subscribing application B is connected to ME3 and receives messages indirectly from ME1, via a subscription on ME2. a remote subscription point on ME 3. These messaging engines are referred to in the following steps.

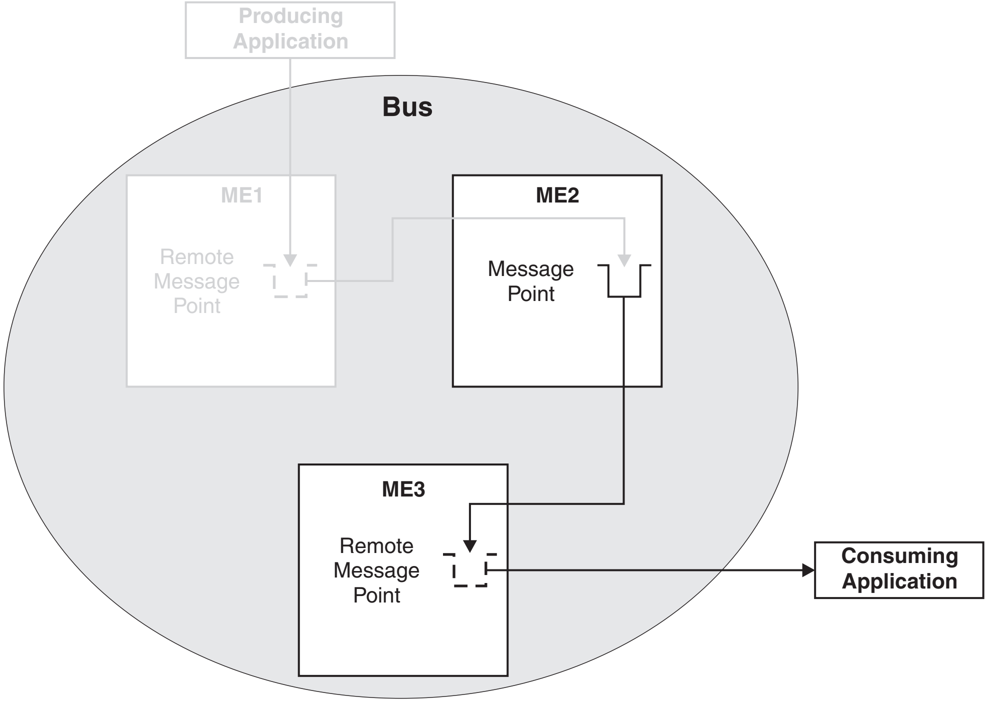

Figure 4. Point-to-point message consumption using a remote message point

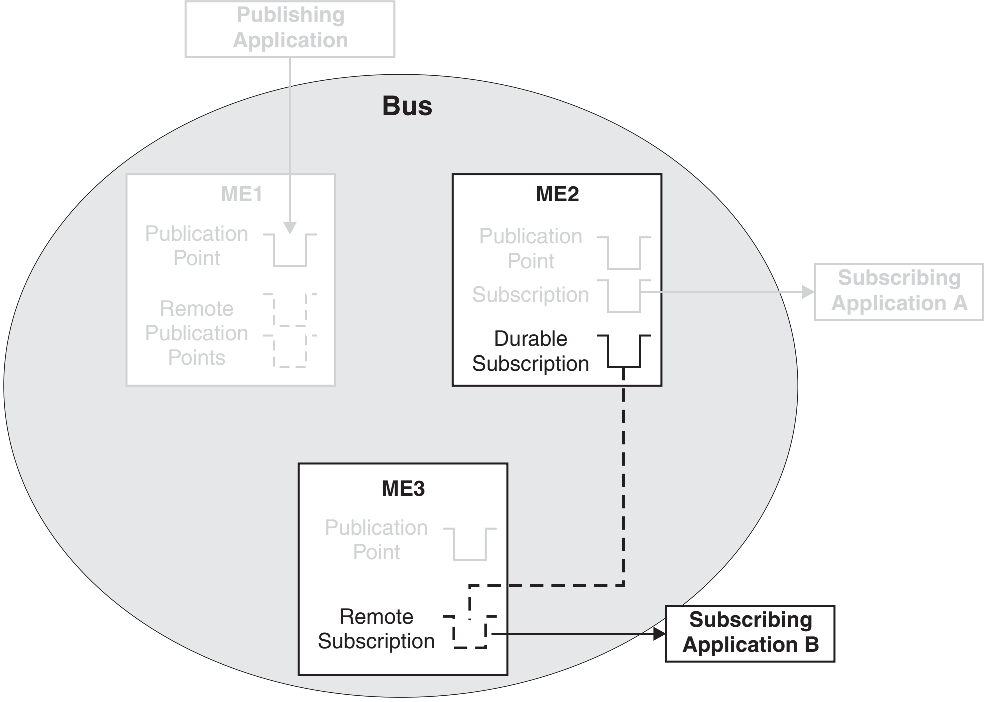

Figure 5. Publish/subscribe messaging using a remote message point

1. If you have followed the steps in “Investigating why point-to-point messages are not being consumed” on page 52 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 59 before starting this task, you should have displayed a list of message requests. On the previous panel (runtime properties for the message point), check that the **Message requests issued** (point-to-point only) or **Message requests received** (publish/subscribe only) value is greater than zero. If the value is not greater than zero, no requests have been made. Check the consuming application for errors:
   - Check that the application is actually connected to ME2.
   - Check that the application did not produce any errors which could explain why messages are not being consumed.
   - Check that the consumer was started.
   - Check that the application did attempt to consume a message:
     - If the application uses an asynchronous consumer, check that the asynchronous consumer was registered.
     - If the application is synchronous, check that the consumer performed a “receive” or a “receive with wait” function (this may require a modification to the application to extend the time that the application waits for a message).
2. If the number of issued message requests is greater than zero, requests from ME3 to ME2 for messages on the message point have been made. Check that the **Completed message requests** value is greater than zero. If not, check that the two messaging engines can communicate with each other, see “Service integration troubleshooting: Checking the communication between two messaging engines in a bus” on page 45.
3. If the number of completed message requests is greater than zero, requests are being issued by ME3, processed by ME2 and completed back to ME3. To ensure that those requests were made by the actual application being investigated, record the current values of **Completed message requests** and either **Message requests issued** or **Message requests received**. Rerun the consuming application and check that both values have increased. If the values do not increase, the application did not make a request from ME3 to ME2 for this message point (the existing numbers relate to a previous application that was consuming messages). Check the consuming application for errors:
   - Check that the application was started.
   - Check that the name of the destination being consumed from is correct.
4. If the values do increase, the message request was issued and completed, but no message was returned or processed by the consuming application.
   - Check that the application’s selection criteria match the available message or messages on the message point.
   - Check that the application is correctly receiving the message, by checking for application or runtime errors.

What to do next

If you are still having problems, contact your IBM customer service representative.

Service integration troubleshooting: Checking the communication between two messaging engines in a bus:

If you are troubleshooting a problem with your service integration system, you might want to check that two messaging engines can communicate with each other.

1. Check that both messaging engines are running.
2. For each messaging engine:
   a. Check the system log for a CWSIT0028I message that relates to the two messaging engines in question. This message indicates that the two messaging engines are successfully communicating with each other.

b. Find the most recent instance (there may be only one) of the CWSIT0028I message for the two messaging engines, and check the log to make sure that a CWSIT0029I message for these two messaging engines does not appear later in the log. This message indicates that the communication connection between the two messaging engines has failed.

If either of these checks fails, inspect the log for indications of the cause of the failure, and follow the suggested actions to rectify the problem.

Investigating why publish/subscribe messages are not arriving at a subscription

This topic describes how to investigate why publish/subscribe messages are not arriving at a subscription on a service integration bus.

About this task

Use this topic if you have an application that is producing messages to a topic space destination, and a consuming application is not receiving the messages.

Perform the following preliminary checks before starting the investigation:

- Check that the producing application is producing messages correctly:
  - Check that there are no failures or runtime errors from the application.
  - Check that the name of the destination is correct.
  - Check that messages are actually being produced.
  - Check that the transaction used to produce the message was committed without any exceptions.
- Check that the consuming application is consuming messages correctly:
  - Check that the application is started.
  - Check that the subscription's topic and selector are correct. Click **Service integration** → **Buses** → ***bus_name*** → **[Destination resources] Destinations** → ***topic_space_name*** → **[Message points] Publication points** → ***publication_point_name*** → **Runtime** → **Subscriptions** → ***subscription_name*** and ensure that the **Topic** and **Selector** fields match the topic and selector specified in the application.
  - If security is enabled, ensure that the subscription has the authority to receive messages sent to it. Refer to Topic security and Messaging security for more information.
- Check the producing application to see if it is giving the messages a short expiry time. If this is the case, the messages may be disappearing before they arrive, or before they can be processed by the receiving messaging engine.
- Click **Service integration** → **Buses** → ***bus_name*** → **[Destination resources] Destinations** to display the destinations on the relevant bus. Click on the topic space and check that the **Send allowed** and **Receive allowed** check boxes are selected.
- Examine the relevant exception destination to see if the messages appear there. If they do, use the information contained within the messages to understand why they have arrived at the exception destination, and write an application (or mediation) to process the messages.
- Check the reliability of the messages. If the reliability is set to best effort, the messages can be discarded by the system during normal operation. See "Understanding why best effort messages are being discarded" on page 43 for a list of possible causes.
- Examine the error logs for exceptions.

1. Click **Service integration** → **Buses** → ***bus_name*** → **[Destination resources] Destinations** to display the destinations on the relevant bus. Click on the relevant topic space and under **Message points**, click **Publication points**. For each publication point listed, click the publication point then click **Runtime >Subscriptions** and look for your subscription. If your subscription is not listed on any of the publication points, there is an error in the consuming application.

2. Determine which messaging engines the producing and consuming applications are connected to. See "Determining which messaging engine an application is connected to" on page 44.
3. If the producing application is connected to the same messaging engine as the consuming application, the messages are being produced locally to the consumer. Recheck the producing and consuming applications, and check the system logs for errors.
4. If the producing application is connected to a different messaging engine than the consuming application, the messages are being routed through a remote publication point. Refer to "Investigating why publish/subscribe messages are not being received by a subscription through a remote message point" to investigate this scenario.

Determining which messaging engine an application is connected to

If your application fails to receive or produce a message, you might want to find out which messaging engine it is connected to, as part of troubleshooting the problem.

1. If your application is a JMS application, examine its connection factory as the messaging engine name might be specified there.
2. If your application is not a JMS application, or its connection factory does not specify the messaging engine name, use one of the following methods to determine which messaging engine the application is connected to:
   - Within the application code, after the application has obtained a valid Connection object, add a call to the toString() method of that object. The connected messaging engine name will be clearly listed when you rerun the application.
   - Enable the SIBJms_External trace component and rerun the application. Inspect the generated trace for a reference to the connected messaging engine name.

Results

Be aware that the messaging engine name returned by either of these methods relates to the rerun of the application. It is possible that the original failing instance of the application was connected to a different messaging engine in the bus.

Investigating why publish/subscribe messages are not being received by a subscription through a remote message point

This topic describes how to investigate why publish/subscribe messages are not being received by a subscription on a service integration bus, when the messages are being routed through a remote message point.

Before you begin

Follow the steps in "Investigating why publish/subscribe messages are not arriving at a subscription" on page 59, which contains preliminary checks and investigative tasks that you should carry out before proceeding with this task.

About this task

Perform this task as part of "Investigating why publish/subscribe messages are not arriving at a subscription" on page 59. This task explains how to investigate the flow of messages in a publish/subscribe messaging scenario where the messages are being routed through a remote message point to a nondurable subscription. The following diagrams illustrates two possible situations. The dotted lines in the diagrams indicate relationships between publication points, whereas the solid lines indicate flow of messages. In Figure 1, ME1 is the messaging engine that the producing application is attached to, and ME2 and ME3 are the messaging engines that are hosting the subscriptions. The publication points on ME2 and ME3 are represented by remote publication points on ME1. In Figure 2, Subscribing application B, which is attached to ME3, has a durable subscription which receives messages through ME2, rather than directly from ME1. In order to do this, ME3 hosts a remote subscription, which represents the subscription on ME2. The messaging engines are referred to in the following steps.

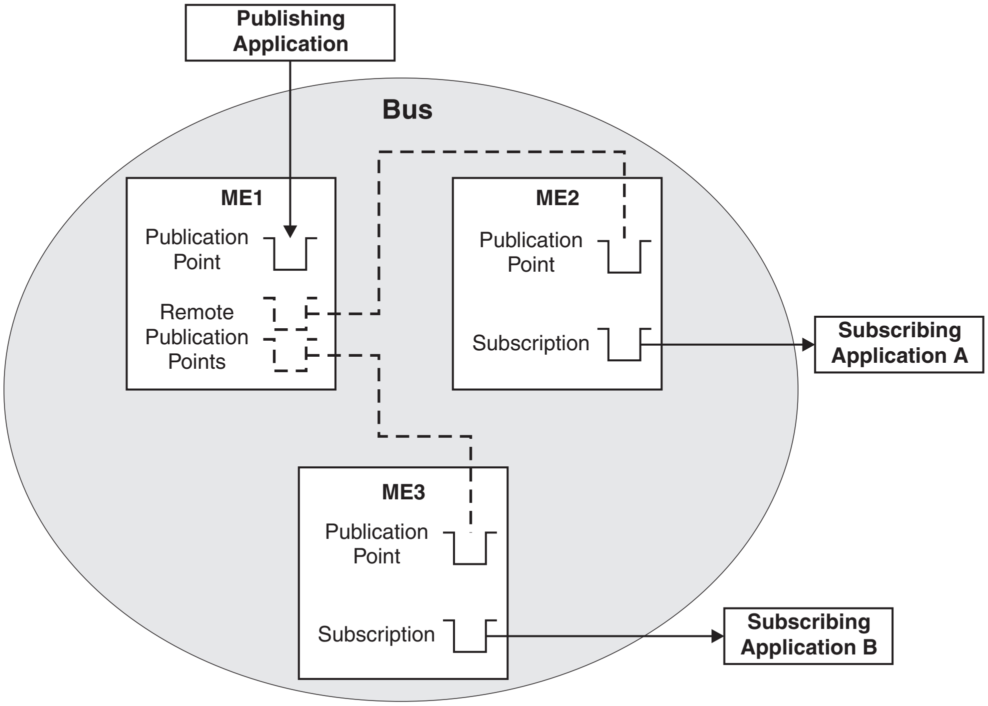

Figure 6. Point-to-point message production using a remote message point

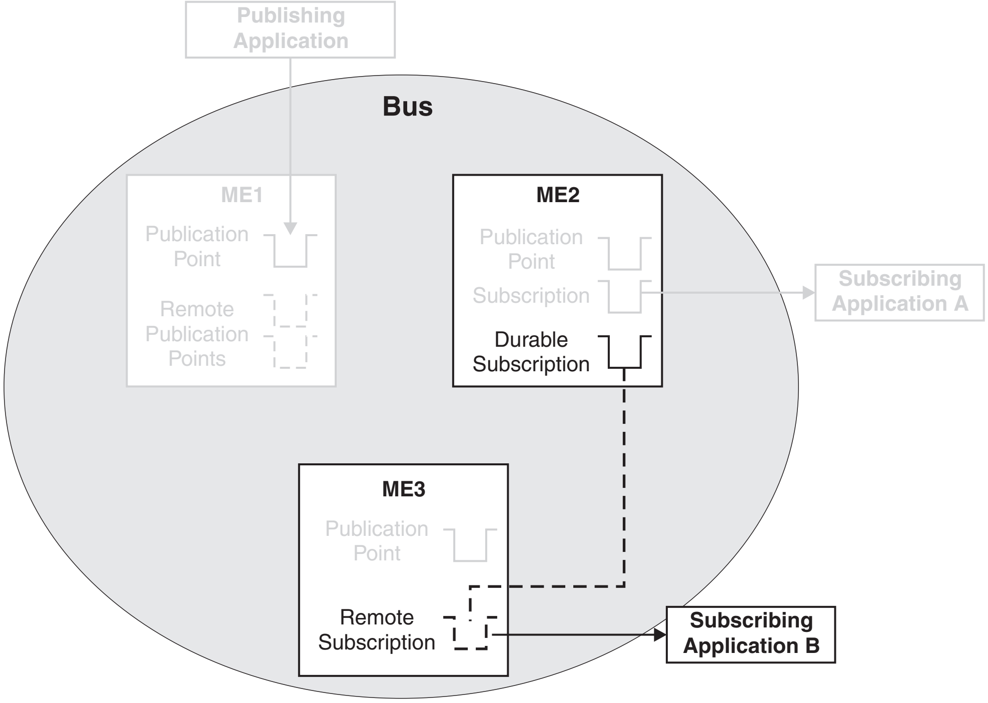

Figure 7. Publish/subscribe messaging using a remote message point

The following steps apply to both the above scenarios.

1. Display the properties for ME1 by clicking **Service integration** → **Buses** → ***bus_name*** → **[Topology] Messaging engines** → ***engine_name***.
2. On the **Runtime** tab for ME1, click **[Remote message points] Remote publication points**, then click the remote publication point that represents the publication point on ME2. Click **Topics** and check that the consumer's topic is listed. If the topic is not listed, perform the following checks:
   - Check that the subscribing application is still running.
   - Check that the two messaging engines can communicate with each other, see "Service integration troubleshooting: Checking the communication between two messaging engines in a bus" on page 45.
   - Wait a short period (dependent on the system configuration) and recheck.
3. It is possible that the registration of the subscription occurred after the message was published. Republish the message and check again to see if it was received.
4. On ME1, again display the remote publication point that represents the publication point on ME2. Review the value of the **Current outbound messages** field.
5. If the number of current outbound messages is greater than zero, messages have been produced but they might not have been received by ME2.
   a. Check that the two messaging engines can communicate with each other, see "Service integration troubleshooting: Checking the communication between two messaging engines in a bus" on page 45.

   b. Look for previous messages on the topic space. If there are previous messages, and some or all of them are for ME2, wait a few moments and then refresh the view.
      - If some of the messages have disappeared from the topic space, the system is currently delivering messages but is backlogged. Wait until the backlog has been cleared, then inspect the publication point on ME2 to see if the test message has arrived.
      - If none of the messages have disappeared from the topic space, the transmission of messages might be blocked by a message that is trapped in the Committing state. Later messages must wait for this message to be delivered, otherwise the ordering of messages will be broken.

        If a message is trapped in the Committing state, that message is contained in an unresolved transaction. A resource manager, such as a database, might have hung. Resolve the issue with the resource manager. If this fails, note the **Transaction ID** of the message and click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **Runtime** → **[Additional Properties] Transaction Service** to display the general properties for the transaction service. Use the **Review** links to resolve the transaction whose **Global ID** matches the transaction ID of the message.

   c. Examine the state of the test message:
      - If the status of the test message is "Pending send", the message is waiting to be sent. ME2 might not be accepting messages. Perform the following checks:
        – Check that the two messaging engines can communicate with each other, see "Service integration troubleshooting: Checking the communication between two messaging engines in a bus" on page 45.

        – Check that the publication point on ME2 is not full: display the runtime properties for the publication point and compare the **Current message depth** to the **High message threshold**. If the current message depth is equal to the high message threshold, the messaging engine will not accept new messages until the queued messages have been consumed. Either restart the consumer and wait until the backlog is cleared, or delete the messages.

        – Check that configuration changes have been propagated. Ensure that ME2 is aware of the existence of the publication point by deploying the latest configuration settings to ME2's application server.

- If the status of the test message is “Pending acknowledgement”, the message has been sent but ME2 has either not received the message, or not processed the message. Check that there are no messages in the Committing state ahead of the test message in the transmit queue, then wait a few moments and examine the publication point again to see if the test message has arrived.If there are messages that are trapped in the Committing state, resolve this problem by referring to the following point.
- If the test message (or another message) is in the Committing state, the message is contained in an unresolved transaction. A resource manager, such as a database, might have hung. Resolve the issue with the resource manager. If this fails, note the **Transaction ID** of the message and click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **Runtime** → **[Additional Properties] Transaction Service** to display the general properties for the transaction service. Use the **Review** links to resolve the transaction whose **Global ID** matches the transaction ID of the message.

6. If the number of completed outbound messages is greater than zero, messages have been produced and processed by ME2, but the test message has not appeared. Rerun the producing application and ensure that the number of completed outbound messages on ME1 increases (you might see the active outbound message count increase before the completed outbound message count increases).
   - If the counts do not increase, the message was not produced at ME1. Check that the producing application is actually connected to this messaging engine (see “Determining which messaging engine an application is connected to” on page 44).
   - If the counts do increase, the message arrived at ME2, but was either consumed, sent to the exception destination, or expired. Check for the presence of consumers, and perform the preliminary checks again.
7. If the number of current and completed messages are both zero, check that the producing application really is producing messages to this destination, by performing the relevant preliminary checks again.
8. You have now checked the flow of messages between ME1 and ME2. If you have an application which is in the position of Subscribing application A or B in Figure 1, you have investigated the situation fully; if you are still having problems, contact your IBM customer service representative. If you have an application which is in the position of Subscribing application B in Figure 2, in other words an application which has a remote subscription, you also need to investigate the flow of messages between ME2 and ME3, using the following steps. To determine if your application is using a remote subscription, display the publication points for the relevant topic space, find your subscription and examine the name of the publication point. The name will be of the form *topic_space_name@messaging_engine_name*. This will tell you which messaging engine the subscription is hosted by. If this messaging engine is different than both the messaging engine that the producing application is connected to, and the messaging engine that the consuming application is connected to, a remote subscription is being used.
9. Display the subscriptions for the publication point on ME2, and find your subscription in the list. If the subscription is not listed, perform the following checks:
   - Check that the subscribing application is still running.
   - Check that the two messaging engines can communicate with each other, see “Service integration troubleshooting: Checking the communication between two messaging engines in a bus” on page 45.
   - Wait a short period (dependent on the system configuration) and recheck.
10. Click your subscription and then click **Known remote subscription points**. In the resulting list, click the name of the messaging engine that is represented by ME3 in the diagram. Click **Message requests**. This displays the requests that have been received by the subscription on ME2, from the remote subscription on ME3.
11. If possible, start the consuming application and ensure that it is actively trying to consume a message (the application should be in either a “receive with wait” state or an “asynchronous consumer registered” state), then follow the instructions in “Investigating why messages are not being consumed through a remote message point or subscription point, while the application is running” on page 53. If your application cannot remain in an actively consuming state for a significant length of time (long

enough to investigate the problem), follow the steps in "Investigating why messages are not being consumed through a remote message point or subscription point, while the application is stopped" on page 56.

Service integration troubleshooting: Checking the communication between two messaging engines in a bus:

If you are troubleshooting a problem with your service integration system, you might want to check that two messaging engines can communicate with each other.

1. Check that both messaging engines are running.
2. For each messaging engine:
   a. Check the system log for a CWSIT0028I message that relates to the two messaging engines in question. This message indicates that the two messaging engines are successfully communicating with each other.
   b. Find the most recent instance (there may be only one) of the CWSIT0028I message for the two messaging engines, and check the log to make sure that a CWSIT0029I message for these two messaging engines does not appear later in the log. This message indicates that the communication connection between the two messaging engines has failed.

   If either of these checks fails, inspect the log for indications of the cause of the failure, and follow the suggested actions to rectify the problem.

Determining which messaging engine an application is connected to:

If your application fails to receive or produce a message, you might want to find out which messaging engine it is connected to, as part of troubleshooting the problem.

1. If your application is a JMS application, examine its connection factory as the messaging engine name might be specified there.
2. If your application is not a JMS application, or its connection factory does not specify the messaging engine name, use one of the following methods to determine which messaging engine the application is connected to:
   - Within the application code, after the application has obtained a valid Connection object, add a call to the toString() method of that object. The connected messaging engine name will be clearly listed when you rerun the application.
   - Enable the SIBJms_External trace component and rerun the application. Inspect the generated trace for a reference to the connected messaging engine name.

Results

Be aware that the messaging engine name returned by either of these methods relates to the rerun of the application. It is possible that the original failing instance of the application was connected to a different messaging engine in the bus.

WS-Notification troubleshooting tips

Tips for troubleshooting your WS-Notification-based publish and subscribe messaging for Web services.

To help you identify and resolve WS-Notification-related problems, use the WebSphere Application Server trace and logging facilities as described in Tracing and logging configuration.

To enable trace for WS-Notification, set the application server trace string to `SIBWsn=all=enabled:com.ibm.ws.sib.webservices.*=all=enabled`. If you encounter a problem that you think might be related to WS-Notification, you can check for error messages in the WebSphere Application Server administrative console, and in the application server `SystemOut.log` file. You can also enable the application server debug trace to provide a detailed exception dump.

A list of the main known restrictions that apply when using WS-Notification is provided in “WS-Notification: Known restrictions” on page 71.

WebSphere Application Server system messages are logged from a variety of sources, including application server components and applications. Messages logged by application server components and associated IBM products start with a unique message identifier that indicates the component or application that issued the message. The prefix for the WS-Notification component is `CWSJN`.

The Troubleshooter reference: Messages topic contains information about all WebSphere Application Server messages, indexed by message prefix. For each message there is an explanation of the problem, and details of any action that you can take to resolve the problem.

Here is a set of tips to help you troubleshoot commonly-experienced problems:
- “Enabling WebSphere Application Server Version 6.1 client applications to handle the additional error conditions introduced in the final Version 1.3 WS-Notification standards”
- “Exception caused by incorrect configuration of the SDO repository” on page 66
- “Multiple messages received by a notification consumer for each event notification” on page 66
- “Deleting administered subscribers and messaging engines” on page 66
- “Use of Web service destinations” on page 67
- “Failure of an inbound (application to broker) notification” on page 68
- “Failure of an outbound (broker to application) notification” on page 68
- “Tidying up stateful resources that are not automatically deleted” on page 69
- “Failure to delete a durable subscription that was created by WS-Notification, when using the service integration bus panel” on page 69
- “Administrative stop of a messaging engine” on page 69
- “Failures as a result of changes in topic space and topic namespace configurations” on page 70
- “Failure to create a WS-Notification service without a configured SDO repository” on page 71

Enabling WebSphere Application Server Version 6.1 client applications to handle the additional error conditions introduced in the final Version 1.3 WS-Notification standards

In WebSphere Application Server Version 6.1, support for WS-Notification is based on a pre-final approval public review draft of the WS-Notification standards. In Version 7.0, this support is extended to cover the final approved standards. The differences between the WS-Notification public review drafts and final standards are as follows:

- • A new fault condition has been added called UnableToGetMessagesFault. It is returned in response to a request to the GetMessages operation if some internal condition means that it is not possible to return messages. Note that this is different to there not being any messages to return, which is handled differently and is the more likely case.
- • The schema for the GetMessages operation no longer requires a value to be passed for the number of messages to return. If no value is passed, then all available messages are returned. This does not affect Version 6.1 client applications, which are already coded to provide a value.
- • The DestroyPullPoint operation now throws the ResourceUnknownFault fault condition in addition to previously declared fault conditions.

The WSDL and schema files shipped with WebSphere Application Server Version 7.0 are updated to reflect the final Version 1.3 WS-Notification standards.

You need not change your existing WS-Notification services. Your existing client applications will also continue to work unchanged, but if they are now also working with new WS-Notification services, and you want them to explicitly handle the new fault conditions, then regenerate the client stubs using the WSDL file from the new service.

Exception caused by incorrect configuration of the SDO repository

If you try to create a WS-Notification service, and you get the following stack trace, then SDO repository is not configured correctly. To resolve this problem, see Installing and configuring the SDO repository.

```
java.lang.Exception: com.ibm.ws.sib.webservices.admin.config.SIBConfigException: CWSWS5010E:
Failed to store WSDL located at http://www.ibm.com/websphere/wsn/notification-broker
due to the following exception: com.ibm.ws.sib.webservices.exception.SIBWSUnloggedException:
CWSWS1007E: The following exception occurred:
com.ibm.ws.sdo.config.repository.impl.RepositoryRuntimeException:
javax.transaction.TransactionRolledbackException: CORBA TRANSACTION_ROLLEDBACK 0x0 No;
nested exception is: org.omg.CORBA.TRANSACTION_ROLLEDBACK:
javax.transaction.TransactionRolledbackException: ; nested exception is:
javax.ejb.TransactionRolledbackLocalException: ; nested exception is:
com.ibm.ws.ejbpersistence.utilpm.PersistenceManagerException:
PMGR1014E: Exception occurred when getting connection factory:
com.ibm.websphere.naming.CannotInstantiateObjectException:
threw NameNotFoundException while the JNDI NamingManager was processing a
javax.naming.Reference object. [Root exception is javax.naming.NameNotFoundException:
Context: smeagolNode03Cell/nodes/smeagolNode03/servers/server1, name:
jdbc/com.ibm.ws.sdo.config/SdoRepository:
First component in name com.ibm.ws.sdo.config/SdoRepository not found.
[Root exception is org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0]] vmcid: 0x0 minor code: 0 completed: No.
```

Multiple messages received by a notification consumer for each event notification

In some situations you might receive more notifications at a given notification consumer than the number of event notifications that have been inserted into the notification broker by a publisher. For example you might publish 4 messages, and receive 8, 12, 16 (or some other multiple of four) messages at the notification consumer.

This is normally caused by there being two or more active subscriptions that target the notification consumer - a situation that can occur if the subscriber application is run more than once. Each time the Subscribe operation is called, a new subscription must be created by the notification broker (see section 4.2 of the Web Services Base Notification specification), which causes duplicate messages to be delivered if a previous subscription exists.

To check whether this is what is happening, examine the **SubscriptionReference** property of the notifications received by the notification consumer. This endpoint reference contains the identifier of the subscription that caused the notification to be sent. If you find several different subscription identifiers, then there is more than one subscription active.

Subscriber applications should tidy up subscriptions when they are not required (or register them with a timeout), however you can tidy them up administratively using the runtime panels as described in Listing or deleting active WS-Notification subscriptions.

Deleting administered subscribers and messaging engines

Deleting administered subscribers

You should be wary of deleting and re-creating messaging engines on bus members for which WS-Notification-administered subscribers have been configured, because in some cases this can leave the remote Web service subscription active (and passing notification messages to the local server) even though there is no longer any record of it.

To avoid this situation you should delete the WS-Notification configuration, or just the administered subscribers, in a separate step to deleting the messaging engine. When the dynamic configuration update is then processed, or the server restarted, the remote Web service subscription is tidied up cleanly.

Note: This problem does not occur if it is only the WS-Notification configuration that is modified; it only occurs if the messaging engine is also deleted.

Administered subscribers in a cluster

When you remove messaging engines from a cluster, remove them in numerical order from highest to lowest so as to avoid a situation where, for example, there are messaging engines numbered 001 and 002 and not 000. This is to avoid problems if you use WS-Notification, which attaches special significance to the first-created messaging engine in a cluster.

In a clustered topology there can be more than one messaging engine running on the "bus member" (cluster). Administered subscribers are defined against a service point (bus member) and so there are several alternatives when choosing the messaging engine that is responsible for creating the subscription to the remote Web service. In this situation, the "first" messaging engine in the cluster is responsible for making the subscription. For example in a cluster containing three messaging engines the messaging engines will have names following the pattern xxx-000-yyy, xxx-001-yyy, xxx-002-yyy, and the administered subscriber subscriptions will be managed by the "000" messaging engine.

If you delete the "000" messaging engine from the cluster then restart the servers, the administered subscriptions are now managed by the "001" messaging engine - being the lowest number engine in the cluster. However, as previously mentioned, deleting and recreating messaging engines on bus members for which administered subscribers have been configured can leave the remote Web service subscription active (and passing notification messages to the local server) even though there is no longer any record of it. Therefore if another messaging engine is later added to the cluster and there is no xxx-000-yyy messaging engine currently defined the new engine takes on the name of xxx-000-yyy. Therefore, in this instance it is possible for two messaging engines to concurrently believe that they manage the administered subscription, resulting in multiple subscriptions being made to the remote Web service.

In the unlikely event that you need to re-create messaging engine xxx-000-yyy, you can avoid duplicate messages from an administered subscription by completing the following steps:

- Delete the administered subscriptions defined to this cluster.
- Allow the existing messaging engines to observe the change. This results in messaging engines in the cluster deleting the administered subscriptions they believe they manage.
- Create the new messaging engine in the cluster.
- Re-create the administered subscriptions.

Use of Web service destinations

Usually, a bus destination can be used as described in Learning about bus destinations. However, this is not the case for WS-Notification. This destination that is associated with the WS-Notification service does not relate to the topics for which the WS-Notification service can handle requests and you should not alter or mediate the destination. In WS-Notification, the configuring of topics is handled through topic namespaces. For more information, see Creating a new WS-Notification permanent topic namespace.

When you create a Version 6.1 WS-Notification service, the wizard configures three service integration bus inbound services for the WS-Notification service, one for each of the three WS-Notification service roles:

- Notification broker
- Subscription manager
- Publisher registration manager

These inbound services are defined on the same service integration bus as the Version 6.1 WS-Notification service, and each of these inbound services refers to the same bus destination.

Failure of an inbound (application to broker) notification

Applications wishing to publish event notifications into the broker make use of the Notify operation. This is defined as a one-way (Web services) operation which means that it is not possible to return a fault (exception) if it is not possible to complete the operation. Thus the application will assume that the notification was successful, but subscribing applications will not receive the notification message.

The cause of this type of failure might be an application error (invalid topic syntax), or a mismatch between the application code and the server configuration (using an undefined topic namespace). Specific reasons for which an inbound notification might fail include the following:

- Invalid topic: the topic expression supplied does not match the syntax of the stated dialect, or they specified an unsupported dialect. This is an application error.
- Invalid topic namespace: the application specifies a topic namespace that has not been configured, but the administrator has specified on the WS-Notification service that we should not permit use of dynamic namespaces. This is caused by a mismatch between the application and the WS-Notification Service topic namespaces.
- Unsupported topic: the specified topic is prohibited by a topic namespace document that has been applied to the topic namespace (for example it is a sub-topic of a topic that has been marked as “final”).
- Invalid credentials: the specified user ID or password is not valid or does not have the required authority. This is caused by a mismatch between the application configuration and the server security policies.
- Publisher not registered: The application tries to send a notification without first registering with the broker, but the administrator has configured the WS-Notification service to require registration of applications. This is caused by an application error - a well behaved application should first check whether it is required to register before publishing to a broker.
- The associated service integration bus topic space has been disabled.

You need to monitor this type of failure closely, because it might indicate a denial of service attack and certainly indicates that the application is not functioning correctly. The first time an inbound notification fails from a particular producing application, a warning message is sent to the SystemOut log of the server. If there are further notification failures for that producer, subsequent timed warning messages are logged at 30 minute intervals. Additional information is provided with each timed message to indicate how many failed notifications were received for that producer during the 30 minute interval.

When the system generates each warning message, it identifies the producing application through one of two identifiers:

- The ProducerReference element of the NotifyMessage provided in the Notify operation . This element uniquely identifies the application. However this element is optional.
- The IP address of the host that originated the request. This address might not uniquely identify the application, but it narrows your search.

Note: The system cannot identify the host IP address in all cases. For example, for SOAP over JMS transports the IP address of the originating host is not available or applicable.

Failure of an outbound (broker to application) notification

The failure of an outbound Web service invocation (broker to application) is caused when a remote application is unavailable for invocation, and might be the result of an application failure, a network error, or a firewall configuration issue. Failure to pass event notifications to subscribed applications causes messages to build up on the subscriptions held on the server. The messages held on a given subscription can be observed using the runtime panels as described in Listing or deleting active WS-Notification subscriptions. Subscriptions for which the most recent event notification attempt has failed in this way are marked as being in **ERROR** state when viewed in the WS-Notification subscription runtime administration panel.

If the WS-Notification service point fails to successfully notify a NotificationConsumer application, a warning message is sent to the application server SystemOut log and the subscription is told to wait for 2 minutes. Reasons for a failure of this type might be that the remote Web service is not currently available, or that network conditions prevent contact between the local server and the service.

After 2 minutes, the notification is retried. If delivery is still not possible then the subscription is put back into a wait state for another 2 minutes. If the failure is caused by a transient I/O error, this pattern is repeated indefinitely, until the notification is either successfully delivered or you delete the subscription. If the error is caused by an application failure on the remote side then the notification will be retried up to the number of times defined in the “Maximum failed deliveries” setting of the service integration bus topic space destination from which the message is being received. After the first warning message is output to the SystemOut log, subsequent timed warning messages are logged at 30 minute intervals.

Tidying up stateful resources that are not automatically deleted

The act of subscribing to the broker or registering a publisher creates a stateful resource on the server that consumes system resources while it is active. Normally an application specifies a termination time as part of the act of creating these resources, and thus they are automatically deleted when the termination time is reached. However it is also possible for the application to request an infinite lifetime for the resource. If this is done then it is possible for resources to remain on the server indefinitely even though the application might never be coming back to use (or destroy) the resource.

You can to view the stateful resources (subscriptions and publisher registrations) using the runtime panels described in Interacting at run time with WS-Notification. These panels also provide the ability to administratively delete the items if required. Only do this if you are sure that the application is no longer using the resource because it will cause application failures if the resource is referenced after being deleted.

Failure to delete a durable subscription that was created by WS-Notification, when using the service integration bus panel

When you create a subscription using a WS-Notification application, that is by using the Subscribe operation, one or more durable subscriptions are created in the relevant service integration bus topic space destination. You can view these durable subscriptions in the service integration bus runtime panels for the publication point.

The runtime panels for the publication point also provide the ability to delete one or more durable subscriptions. However, if you use this function to delete a subscription that was created by a WS-Notification application, the delete operation fails. This failure occurs because the WS-Notification implementation maintains an active consumer for this durable subscription for the duration of the time that the server is running, and a durable subscription cannot be deleted if an active consumer is present.

Note: This deletion restriction also applies to durable subscriptions created by other applications, such as JMS applications.

To delete a subscription that was created by a WS-Notification application, use the runtime panels provided by the WS-Notification implementation, as described in Interacting at run time with WS-Notification. This approach closes the active consumer and automatically deletes the related service integration bus durable subscriptions.

Administrative stop of a messaging engine

WebSphere Application Server depends on being able to access a running service integration bus messaging engine to send and receive messages, and to create and retrieve state for the various Web service resources that are created.

You can stop a messaging engine using the MBean interface or runtime panels. This prevents WS-Notification from successfully servicing any requests from applications that might come in during the time that the messaging engine is stopped. In this situation, error messages are logged as described in "Failure of an inbound (application to broker) notification" on page 68 and "Failure of an outbound (broker to application) notification" on page 68. When you stop a messaging engine, all WS-Notification processing stops and all messaging applications cease to function. When you restart the messaging engine, WS-Notification processing resumes.

Failures as a result of changes in topic space and topic namespace configurations

The WS-Notification configuration artifacts often depend on objects defined in other areas of the server configuration. For example the endpoint listeners through which application requests are received, and the service integration bus topic spaces to and from which messages are sent.

The following items describe the action that is taken by the WS-Notification runtime code when it meets relevant changes in the objects upon which it depends.

Deleting a service integration bus topic space

The service integration bus topic space is the primary messaging object upon which WS-Notification depends at run time. Notification messages from an application are published to the topic space specified by the (permanent) topic namespace mapping specified by the administrator.

Deleting a service integration bus topic space has the following effects upon new and existing WS-Notification applications:

- RegisterPublisher requests using a WS-Notification topic namespace that references the deleted topic space receive a TopicNotSupportedFault error message.
- Notify requests for a topic associated with the deleted topic space do not publish the message to the topic space (because it has been deleted). The application is not informed because no faults are thrown by the Notify operation (see "Failure of an inbound (application to broker) notification" on page 68).
- Subscribe requests using a WS-Notification topic namespace that references the deleted topic space receive a SubscribeCreateFailedFault error message.
- No further messages are delivered to applications that have existing subscriptions to the deleted topic space. The existing subscription is deleted, and any attempt to invoke operations on the subscription (for example getCreationTime) results in a ResourceUnknownFault error message.
- Deleting and recreating a service integration bus topic space is considered as two separate steps. Existing subscriptions are deleted in response to the first step, and therefore do not exist when the topic space is recreated.

Deleting a permanent topic namespace mapping

Deleting the topic namespace mapping that was used to establish a (currently active) subscription has the same effect as deleting the underlying service integration bus topic space as defined previously, and subscriptions that were created using this namespace mapping are deleted.

Publisher registrations and pull points associated with the deleted topic namespace mapping are also deleted.

Changing a permanent topic namespace mapping

The fields of a permanent topic namespace mapping are read-only fields, so the only way to "change" the fields is to delete the namespace mapping and recreate it with new values. The effect of deleting a permanent topic namespace mapping is described in the previous item.

Failure to create a WS-Notification service without a configured SDO repository

When you create a WS-Notification service, WSDL documents are saved into the SDO repository. You will see the following error message if you try to create a WS-Notification service by using the administrative console, or through scripting, before successfully configuring the SDO repository.

```
java.lang.Exception: com.ibm.ws.sib.webservices.admin.config.SIBConfigException: CWSWS5010E: Failed to
store WSDL located at http://www.ibm.com/websphere/wsn/notification-broker due to the following
exception:
    com.ibm.ws.sib.webservices.exception.SIBWSUnloggedException: CWSWS1007E: The following exception
        occurred: com.ibm.ws.sdo.config.repository.impl.RepositoryRuntimeException:
        javax.transaction.TransactionRolledbackException: CORBA TRANSACTION_ROLLEDBACK 0x0 No; nested
        exception is: org.omg.CORBA.TRANSACTION_ROLLEDBACK:
        javax.transaction.TransactionRolledbackException: ;
    nested exception is: javax.ejb.TransactionRolledbackLocalException: ;
    nested exception is: com.ibm.ws.ejbpersistence.utilpm.PersistenceManagerException: PMGR1014E:
        Exception occurred when getting connection factory:
        com.ibm.websphere.naming.CannotInstantiateObjectException: threw NameNotFoundException while
        the JNDI NamingManager was processing a javax.naming.Reference object.
    [Root exception is javax.naming.NameNotFoundException: Context:
        KADGINNode01Cell/nodes/KADGINNode01/servers/server1, name:
        jdbc/com.ibm.ws.sdo.config/SdoRepository: First component in name
        com.ibm.ws.sdo.config/SdoRepository not found.
    [Root exception is org.omg.CosNaming.NamingContextPackage.NotFound:
        IDL:omg.org/CosNaming/NamingContext/NotFound:1.0]] vmcid: 0x0 minor code: 0 completed: No.
```

For details on how to configure the SDO repository, see Installing and configuring the SDO repository.

WS-Notification: Known restrictions

The main known restrictions that apply when using WS-Notification.

- “Composition with WS-Policy”
- “Optional specification elements”
- “Interpretation of the specification” on page 72

Composition with WS-Policy

This implementation of WS-Notification does not compose with WS-Policy.

Optional specification elements

The WS-Notification standards define a series of optional elements that can be implemented at the discretion of the provider. The following items list those optional elements that are supported or not supported in WebSphere Application Server:

Supported optional elements

All three topic dialects that are defined by the WS-Topics standard are supported in WebSphere Application Server:

- *Simple topics*. That is, single-level root topics with no wildcards. For example “stock”.
- *Concrete topics*. That is, multi-level topics with no wildcards. For example “stock/IBM”, “sport/football/results”.
- *Full topics*. That is, multi-level topics with wildcards and conjunctions. For example “stock//.”, “sport/football/*”, “sport/*/results”, “t1/t3 | t3/t4”.

Subscription and PublisherRegistration termination is supported. That is, scheduled and immediate destruction of WS-Resources.

RequiresRegistration is supported, and can be set to “true” or “false”.

Demand-based publishers, as defined in Chapter 4 of the brokered notification specification, are supported. Demand based publishers allow producers to request that they be paused or resumed by the broker, depending upon whether there are any consumers listening on the topics for which they produce messages. This supports situations where it is expensive to create a notification message.

Unsupported optional elements

The following optional operations from WS-ResourceProperties for SubscriptionManager and PublisherRegistrationManager are not supported:

- GetMultipleResourceProperties
- SetResourceProperties
- QueryResourceProperties
- GetResourcePropertyDocument.

Consequently, after a subscription is created, only its WS-ResourceProperties ResourceLifetime scheduled destruction properties can be modified.

Filtering of the following event notifications (selectors) is not supported:

- The XPath 1.0 dialect as specified in the XML Path Language (XPath) Version 1.0 W3C recommendation, where the evaluation context is the NotificationMessage.
- The XPath 2.0 dialect.
- Any filter defined as executed over the message body.

Calling the GetCurrentMessage operation always results in a NoCurrentMessageOnTopicFault exception.

Interpretation of the specification

There are several areas of the WS-Notification standards in which decisions are left open to the implementor, or not fully specified. The following items describe the interpretations made in this implementation.

Messages that are published while a subscription is paused

The Web Services Base Notification specification describes several options that are open to the implementor regarding what to do with messages that are generated by a NotificationProducer (or NotificationBroker) while a subscription is paused. In this implementation all notifications that are generated during the period of time a subscription is paused are retained at the server until the subscription is resumed.

Lifetime of a pull point that has been associated with a subscription

A pull point that has been associated with a subscription remains in existence when the associated subscription is deleted. However any calls to GetMessages for that pull point return zero messages.

Conversely, if a pull point associated with a subscription is deleted or expired then the associated subscription remains in existence. However you cannot get any messages from it, and you cannot associate an existing subscription with a new pull point.

Chapter 4. Data access resources

Using a single instance of a resource adapter

You can restrict certain resource adapters to a single runtime instance inside the JVM.

Before you begin

Enabling this setting imposes a highly restrictive environment on the system and should be used with caution.

About this task

Using the single-instance resource adapter configuration option on some resource adapters can enable you to set up an environment that optimally behaves. Some resource adapters that support inbound communications from the enterprise information system (EIS) might require single-instance behavior. By enabling this setting, server startup time can be optimized. Other resource adapters might not require this setting. You need to determine if you should configure the resource adapter for single-instance behavior.

Consider using the single-instance resource adapter configuration for testing and troubleshooting problems. Placing the single-instance restriction on some resource adapters might work as a corrective action for problems; enabling single-instance behavior on one or more resource adapters thought to be involved in a problem can help isolate the specific issue.

This design does not allow two resource adapter JavaBeans instances that would return true from the equals method to coexist in the same JVM, if any one of them is configured as single-instance. For example, if two applications that have embedded the same resource adapter, or one application that embeds a resource adapter and the same resource adapter is installed in the server as a standalone resource adapter, are configured on the same server such that even though some of their config attributes are different, the ones that the equals() method evaluates are equal, this will no longer be allowed, and will return a ResourceException.

Note: The vendor of a resource adapter which cannot tolerate multiple instances does not have a JCA-defined method of communicating this. Therefore, it is up to the deployer to recognize the need, and configure resource adapter(s) for single-instance behavior.

Chapter 5. Messaging resources

Troubleshooting WebSphere messaging

Use this overview task to help resolve a problem that you think is related to the WebSphere Messaging.

About this task

To identify and resolve problems that you think are related to WebSphere Messaging, you can use the standard WebSphere Application Server troubleshooting facilities. If you encounter a problem that you think might be related to WebSphere Messaging, complete the following stages. Some problems and their troubleshooting are specific to whether you are using the default messaging provider, or WebSphere MQ as the JMS provider.

1. Check for error messages about messaging:
   a. Check for error messages that indicate a problem with JMS resources.

      Check in the application server's SystemOut log at *was_home*\logs*server*\SystemOut.

      The associated message reference information provides an explanation and any user actions to resolve the problem. For details, see Troubleshooter reference: Messages in the information center.

   b. Check for other informational and error messages that might provide a clue to a related problem. For example, if you have problems accessing JMS resources, check for more error messages and extra details about any problem associated with the JMS provider or with the service integration technologies that the default messaging provider uses.

      For messages related to the resource adapter (JMS) of the default messaging provider, look for the prefix: CWSJR. For messages related to service integration technologies, see the related reference topics.
2. If you suspect that problems might be related to use of message-driven beans, see "Troubleshooting message-driven beans" on page 81.

   If your message-driven bean uses WebSphere Application Server Version 5 JMS resources, look for the prefixes: MSGS and WMSG.
3. If you are using the default messaging provider, use the following administrative console panels to inspect the configuration of your applications and JMS resources:
   - For a view of the JMS resources for a given application, see the following panel: Messaging resources for this application.
   - For a view of the applications and JMS resources for a given default messaging provider destination, see the following panel: Application resources for this destination.
4. Check the Release Notes for specific problems and work arounds. The section *Possible Problems and Suggested Fixes* of the Release Notes, available from the WebSphere Application Server library web site, is updated regularly to contain information about known defects and their workarounds. Check the latest version of the Release Notes for any information about your problem. If the Release Notes do not contain any information about your problem, you can also search the Technotes database on the WebSphere Application Server web site.
5. Check your JMS resource configurations If the messaging services seem to be running properly, check that the JMS resources have been configured correctly. For example, check that the JMS activation specification against which a message-driven bean is deployed has been configured correctly. For more information about configuring JMS resources, refer to the topic "Choosing a JMS provider for your messaging application".
6. Get a detailed exception dump for messaging. If the information obtained in the preceding steps is still inconclusive, you can enable the application server debug trace for the ″Messaging″ group to provide a detailed exception dump.

Messaging troubleshooting tips

These tips are to help you troubleshoot your WebSphere messaging configuration.

To help you identify and resolve problems with messaging, you can use the WebSphere Application Server trace and logging facilities as described in Tracing and logging configuration.

If you are having problems deploying or running applications that use the WebSphere Application Server messaging capabilities, see the following topics:

- “Troubleshooting WebSphere messaging” on page 75
- “Troubleshooting message-driven beans” on page 81
- Troubleshooting service integration technologies
- Default messaging provider: Troubleshooting tips

If you see WebSphere MQ error messages or reason codes in WebSphere Application Server messages and logs, refer to the WebSphere MQ Messages document.

Check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

Here is a set of tips to help you troubleshoot commonly-experienced problems. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

- “A JMS application can no longer send or receive messages, or a destination becomes full and can no longer receive messages.”
- “javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log ” on page 77
- “javax.jms.JMSException: MQJMS3024: unable to start MDB listener” on page 77
- “SVC: jms.BrokerCommandFailedExceptfailed: 3008” on page 77
- “An MDB listener fails to start” on page 77
- “Problems running JMS applications with security enabled” on page 78
- “Messaging troubleshooting tips”
- “Server memory consumption and java.lang.OutOfMemoryError exception when processing JMS messages” on page 78
- “TopicConnectionFactory attributes clash error when using ″Basic″ WebSphere MQ broker (MA0C SupportPac broker)” on page 79
- “WSEC5061E: The SOAP Body is not signed exception is issued when running a secured Web services application using JMS transport and WebSphere MQ” on page 79
- “Message MQJMS1006: invalid value for tempQPrefix is issued when trying to use a Version 5.1 client with a V5 default messaging provider queue connection factory on a Version 7 application server” on page 79
- “When you use WebSphere MQ as an external JMS provider, messages sent within a user-managed transaction arrive before the transaction commits” on page 80
- “javax.jms.JMSException: MQJMS3024: unable to start MDB listener” on page 80
- “javax.jms.JMSException: MQJMS3024: unable to start MDB listener” on page 77

A JMS application can no longer send or receive messages, or a destination becomes full and can no longer receive messages.

When you configure an application to use the default messaging provider, you associate it with either of the following resource sets:

- One or more message beans connected through Java Message Service (JMS) activation specifications.
- One or more enterprise beans connected through JMS connection factories and JMS destinations.

To help resolve this problem, use the following administrative console panels to inspect the configuration of your applications and JMS resources:

- For a view of the JMS resources for a given application, see the following panel: Messaging resources for this application.
- For a view of the applications and JMS resources for a given default messaging provider destination, see the following panel: Application resources for this destination.

javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log

This error can occur when the WebSphere MQ queue name is not defined in the internal Java Message Service (JMS) server queue names list. This problem can occur if a WebSphere Application Server queue destination is created without adding the queue name to the internal JMS server queue names list.

To resolve this problem:
1. Start the WebSphere Application Server administrative console.
2. Navigate to **Servers** → **Manage Application Servers** → ***server_name*** → **Server Components** → **JMS Servers**
3. Add the queue name to the list.
4. Save the changes and restart the server.

javax.jms.JMSException: MQJMS3024: unable to start MDB listener

This error can occur if an uninitialized client ID is used (that is a client ID that is not associated with a durable subscription).

To resolve this problem, set the client ID in *one* of the following three ways:
- As a property of the tcf via jmsadmin (for example alter tcf(myTCF) clientid(myID))
- Programmatically using TopicConnection.setClientID()
- Through the client ID field on the WebSphere MQ topic connection factory resource. For more information, see “WebSphere MQ connection and queue connection factory creation errors” on page 80.

SVC: jms.BrokerCommandFailedExceptfailed: 3008

One possible cause for this error is that you logged on to a Windows® 2000 system as an administrator.

To resolve this problem, log out and log in again as a user, rather than an administrator.

An MDB listener fails to start

If an MDB listener deployed against a listener port fails to start, you should see the following message:

```
WMSG0019E: Unable to start MDB Listener {0}, JMSDestination {1} : {2}
```

To help resolve this problem, check the following factors:
- Check that the administrative resources have been configured correctly. For example, use the administrative console to check the listener port properties: Destination JNDI name and Connection factory JNDI name. Check that other properties of the listener port, destination, and connection factory are correct.
- Check that the queue exists and has been added to the JMS server.
- Check that the queue manager and JMS server have started.
- Check that the Remote Queue Manager Listener has started.
- If security is enabled, check that a component-managed authentication alias has been specified on the queue connection factory or topic connection factory used by the message-driven bean. For more information, see “Problems running JMS applications with security enabled” on page 78.

Problems running JMS applications with security enabled

When you try to run a JMS application with security enabled, you can encounter authentication problems indicated by one or more of the following error messages:

```
WMSG0019E: Unable to start MDB Listener PSSampleMDB, JMSDestination Sample/JMS/listen : javax.jms.JMSSecurityException:
```

This example indicates that the security credentials supplied are not valid.

To resolve this problem, check the security configuration:

- If the authentication mechanism is set to `Application`, the application must supply valid credentials.
- If the authentication mechanism is set to `Container`, you must configure the JMS connection factory with a container-managed authentication alias, and ensure that the associated user ID and password are valid.

```
MQJMS2013 invalid security authentication supplied for MQQueueManager:
```

If you are using WebSphere MQ as a JMS provider, with JMS connection using bindings transport mode, and the user specified is not the current logged-on user for the WebSphere Application Server process, the JMS bindings authentication by WebSphere MQ generates an invalid security authentication error.

To resolve this problem, check the security configuration. When you configure the WebSphere MQ JMS provider to use bindings transport mode, you set the property **Transport type** to BINDINGS on the WebSphere MQ queue connection factory. At this time, you also choose one of the following options:

- Use security credentials. To do this, ensure that the user specified is the currently logged-on user for the WebSphere Application Server process.
- Do not use security credentials. On the WebSphere MQ connection factory, ensure that the **Component-managed Authentication Alias** and the **Container-managed Authentication Alias** properties are not set.

For more information about messaging security, see the Asynchronous messaging - security considerations section of the *Developing and deploying applications* PDF book.

Server memory consumption and java.lang.OutOfMemoryError exception when processing JMS messages

When you use the default messaging provider, JMS messages are processed by a messaging engine within the application server process. This approach consumes memory from the application server's JVM heap. If there is significant concurrent processing of large messages, and the amount of memory available to the JVM heap is not enough to handle this event, then a java.lang.OutOfMemoryError exception is thrown and the application server terminates.

To resolve this problem, estimate the potential number of concurrent processors or consumers of messages and the message sizes, then set the size of the application server's JVM heap to handle the effect. For example:

1. When you deploy a message-driven bean that processes messages concurrently, estimate the potential consumption of the application server's memory by concurrent endpoints. Note that each endpoint that is concurrently processing a message request adds at least two times the message size to the server's JVM heap and can add more, especially if a two-phase transaction is in place.
2. Start the WebSphere Application Server administrative console.
3. Navigate to **Servers** → **Application servers** → ***server_name*** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**, then configure the amount of memory available to the application server's JVM heap by setting the **Initial Heap Size** and **Maximum Heap Size** properties.
4. Navigate to **Resources** → **JMS** → **JMS providers** → **Default messaging provider** → **Activation specifications** → ***activation_specification_name***, then configure the number of concurrent MDB endpoints that can process messages by setting the **Maximum concurrent endpoints** property of the activation specification for this message-driven bean.

TopicConnectionFactory attributes clash error when using ″Basic″ WebSphere MQ broker (MA0C SupportPac™ broker)

When you create a JMS topic subscriber that uses the WebSphere MQ messaging provider, the following error message can occur in the SystemOut.log file:

```
WSVR0017E: Error encountered binding the J2EE resource, TopicConnectionFactory, as <JNDI_NAME>
   from file:<RESOURCES_FILE> com.ibm.ws.runtime.component.binder.ResourceBindingException: invalid
   configuration passed to resource binding logic. REASON: Failed to create connection factory:
   Error raised constructing AdminObject, error code:  TopicConnectionFactory attributes clash  :
   TopicConnectionFactory attributes clash
```

This problem is caused by the configuration of the JMS topic connection factory that is used to create the subscriber, which specifies a broker version of ″Basic″ and a message selection value of ″Broker″. The ″Basic″ WebSphere MQ broker (MA0C SupportPac broker) does not support ″Broker″ message selection.

To resolve this problem, change the JMS topic connection factory to specify a message selection value of ″Client″, which is the only supported value for the WebSphere MQ Basic broker (MA0C SupportPac broker).

“WSEC5061E: The SOAP Body is not signed” exception is issued when running a secured Web services application using JMS transport and WebSphere MQ

When you run a secured Web services application that uses JMS transport under the WebSphere MQ messaging provider, the following error message can occur in the SystemOut.log file:

```
com.ibm.wsspi.wssecurity.SoapSecurityException: WSEC5061E: The SOAP Body is not signed.; null
```

This problem occurs under the following circumstances:

- A Web service application, configured with Web services security, is running in an application server that has WebSphere Application Server security enabled.
- This Web service application uses the JMS transport to send SOAP requests to a target Web service.
- The JMS resource uses a remote WebSphere MQ server to connect to a WebSphere MQ queue.
- Another identical Web service application, configured to use the same queue through the same WebSphere MQ server, is running in a different application server that does not have WebSphere Application Server security enabled.

The problem occurs when a request sent from the original application is processed through the same queue, but to the different application server where security is not enabled.

To resolve this problem:

1. Create a unique queue manager with a unique port in the WebSphere MQ server.
2. Reconfigure the JMS resources to use the new queue manager and port; for example, by using the WebSphere Application Server administrative console to change the properties of the WebSphere MQ queue connection factory. For information about how to perform this task, read the section, Configuring a JMS queue connection factory for WebSphere MQ in the *Administering applications and their environment* PDF book.
3. Rerun the application.

Message “MQJMS1006: invalid value for tempQPrefix” is issued when trying to use a Version 5.1 client with a V5 default messaging provider queue connection factory on a Version 7 application server

When you use a WebSphere Application Server Version 5.1 application client to connect to a queue connection factory defined as a “V5 default messaging provider”″ resource on a WebSphere Application Server Version 7 application server, the following message is displayed:

```
com.ibm.websphere.naming.CannotInstantiateObjectException: Exception occurred while the JNDI
   NamingManager was processing a javax.naming.Reference object.
Root exception is com.ibm.websphere.naming.CannotInstantiateObjectException: Exception occurred
   while the JNDI NamingManager was processing a javax.naming.Reference object.
Root exception is javax.jms.JMSException: MQJMS1006: invalid value for tempQPrefix:
```

This problem occurs when the application client is using WebSphere MQ JMS client CSD 04 JAR files. WebSphere Application Server Version 7 sets the **tempQprefix** property to blank, and this value cannot be handled by the CSD 04 release of the setTempqPrefix method.

To resolve this problem:

- If the application client uses WebSphere embedded messaging JAR files, apply the WebSphere embedded messaging interim fixes for WebSphere Application Server Version 5.1.
- If the client uses external WebSphere MQ JMS client JAR files, apply the CSD 05 release.

When you use WebSphere MQ as an external JMS provider, messages sent within a user-managed transaction arrive before the transaction commits

When you use WebSphere MQ as an external JMS provider, and you send a message to a WebSphere MQ queue within a user-managed transaction, the message can arrive on the destination queue before the transaction commits. This problem occurs when the WebSphere MQ resource manager is not enlisted in the user-managed transaction.

To resolve this problem, use a container-managed transaction.

javax.jms.JMSException: MQJMS3024: unable to start MDB listener

This error can occur if you use an uninitialized client ID (that is, a client ID that is not associated with a durable subscription). To resolve this problem, set the client ID in *one* of the following three ways:

- Set the client ID as a property of the tcf, using the jmsadmin tool. For example, alter `tcf(myTCF) clientid(myID)`.
- Set the client ID programmatically, using `TopicConnection.setClientID()`
- Set the client ID field administratively, using the administrative console to modify the WebSphere MQ messaging provider topic connection factory settings.

SVC: jms.BrokerCommandFailedExceptfailed: 3008

This error can occur when you log in to a Windows 2000 system as an administrator.

To resolve this problem, log out and log in again as a user, rather than an administrator.

WebSphere MQ connection and queue connection factory creation errors

You can receive exception errors when trying to create a MDBListener instance, because the MQ manager userid does not have write access to the `/tmp` directory.

If this problem does not resembles yours, or if the information provided does not solve your problem, see Troubleshooting WebSphere Messaging. If you are still unable to resolve the problem, contact IBM support for further assistance.

The following exception may occur when trying to create the MDBListener instance:

```
6/23/03 22:45:58:232 CDT] 673106a8 MsgListenerPo W WMSG0049E:
Failed to start MDB PSSampleMDB against listener port SamplePubSubListenerPort
[6/23/03 22:47:58:289 CDT] 673106a8 FreePool E J2CA0046E:
Method createManagedConnctionWithMCWrapper caught an exception
during creation of the ManagedConnection for resource
JMS$SampleJMSQueueConnectionFactory, throwing ResourceAllocationException.
Original exception:  javax.resource.spi.ResourceAdapterInternalException:
 createQueueConnection failed
```

```
com.ibm.mq.MQException: MQJE001: An MQException occurred:
Completion Code 2, Reason 2009
MQJE003: IO error transmitting message buffer at
com.ibm.mq.MQManagedConnectionJ11.(MQManagedConnectionJ11.java:239)
```

This problem occurs because the MQ manager userid does not have write access to the `/tmp` directory. To correct this problem, before you use a Jacl procedure to configure WebSphere Application Server resources and install an application:

1. Ensure that all applications have write access to /tmp directory. Use the chmod 1777 command on the directory if necessary.
2. Create another subdirectory under `/tmp` (for example, `/tmp/mydir`). Use this directory as a ″working directory″ for the Jacl.
3. Restart the server.

Applications that use messaging on startup should start successfully.

Troubleshooting message-driven beans

Use this overview task to help resolve a problem that you think is related to message-driven beans.

About this task

Message-driven beans support uses the standard WebSphere Application Server troubleshooting facilities. If you encounter a problem that you think might be related to the message-driven beans, complete the following steps.

1. Check for error messages about message-driven beans:
   a. Check for error messages that indicate a problem with JMS resources, such as activation specifications or listener ports, that are used by message-driven beans.

      Check in the application server’s SystemOut log at *was_home*\logs*server*\SystemOut.

      The associated message reference information provides an explanation and any user actions to resolve the problem. For details, see Troubleshooter reference: Messages in the information center.

   b. Check for more informational and error messages that might provide a clue to a related problem. For example, if you have problems accessing JMS resources, check for more error messages and extra details about any problem associated with the JMS provider or with the service integration technologies that the default messaging provider uses.

      For messages related to the resource adapter (JMS) of the default messaging provider, look for the prefix: CWSJR. For messages related to service integration technologies, see the related reference topics.

      If your message-driven bean uses WebSphere Application Server Version 5 JMS resources, look for the prefixes: MSGS and WMSG.
2. If you are using the default messaging provider, use the following administrative console panels to inspect the configuration of your message-driven beans:
   - For a view of the JMS resources for a given message-driven bean, see the following panel: Messaging resources for this application.
   - For a view of the message-driven beans and JMS resources for a given default messaging provider destination, see the following panel: Application resources for this destination.
3. Check the Release Notes for specific problems and workarounds. The section *Possible Problems and Suggested Fixes* of the Release Notes, available from the WebSphere Application Server library Web site, is updated regularly to contain information about known defects and their workarounds. Check the latest version of the Release Notes for any information about your problem. If the Release Notes does not contain any information about your problem, you can also search the Technotes database on the WebSphere Application Server Web site.
4. If your message-driven bean is deployed against a listener port, check that the listener port has started. The message listener service is an extension to the JMS functions of the JMS provider. For

each message-driven bean mapped to a listener port, the message listener service controls a listener which monitors a JMS destination on behalf of a deployed message-driven bean.

5. Check your JMS resource configurations. If the messaging services seem to be running properly, check that the JMS resources have been configured correctly. For example, check that the JMS activation specification against which the message-driven bean is deployed has been configured correctly. For more information about configuring JMS resources for message-driven beans, see Administering support for message-driven beans in the *Administering applications and their environment* PDF book.
6. Get a detailed exception dump for messaging. If the information obtained in the preceding steps is still inconclusive, you can enable the application server debug trace for the ″Messaging″ group to provide a detailed exception dump.

Troubleshooting performance monitoring statistics

Use this task to resolve inconsistencies between performance monitoring statistics for the enterprise bean counters MethodLevelCallCount and MessageCount when deploying message driven beans in a cluster environment.

About this task

This task addresses inconsistencies in performance monitoring statistics for message driven beans in a cluster environment. Sometimes the number of messages in a trace correspond with the Performance Monitoring Infrastructure (PMI)/Tivoli Performance Viewer (TPV) output statistics but not with the log message from the message driven bean. This arises because the MethodLevelCallCount enterprise bean counter has a different meaning for message driven beans than for other beans.

In general, in terms of PMI statistics collection, and with reference to the Enterprise JavaBeans (EJB) container, message delivery comprises the following steps:

1. EJB container pre-invoke processing. This prepares the execution environment for message delivery.
2. Removing the message from the queue and invoking the message driven bean method to process that message.
3. EJB container post-invoke processing. This cleans up the execution environment for example, committing or rolling back the transaction started during pre-invoke processing.

If there are multiple servers or threads attempting to remove a message from the queue and deliver it to a message driven bean, at Step 2 a messaging service may discover that the queue is empty and there is nothing to deliver because another server or thread has already processed the message. If this happens, the message driven bean method is not called at Step 2 and therefore the MethodLevelCallCount does not correspond to the number of times a message is delivered to the message driven bean for processing. Instead, the MethodLevelCallCount indicates the number of times message delivery is attempted: the enterprise bean counter MessageCount indicates the number of successful deliveries to a message driven bean.

In a single server environment MethodLevelCallCount and MessageCount should be the same. However, in a multi-server environment (or an environment with multiple consumers), these counts can be different. If the difference is large, consider retuning your messaging system.

To identify and resolve problems that you think are related to WebSphere Messaging, you can use the standard WebSphere Application Server troubleshooting facilities. If you encounter a problem that you think might be related to WebSphere Messaging or to other message-driven beans issues, see the related links.

Chapter 6. Mail, URLs, and other J2EE resources

Debugging a mail session

When you need to debug a mail application, you can use the mail debugging feature. The mail component will generate debugging information, on a per session basis, that can be used for problem determination or tuning.

About this task

Enabling the debug mode triggers the mail component of the application server to print the following data to the standard output stream:

- interactions with the mail servers
- properties of the mail session

This output stream is redirected to the SystemOut.log file for the specific application server.

1. Open the administrative console.
2. Click **Resources** → **Mail** → **Mail sessions** → ***mail session***.
3. Click **Enable debug mode**. Debugging is enabled for that session only.
4. Click **Apply** or **OK**.

Example

The following example shows sample mail debugging output:

```
ResourceMgrIm I   WSVR0049I: Binding Test as mail/test
SystemOut     O *** In SessionReferenceable.getReference:
SystemOut     O  added StringRefAddr: type=ws.transport.password, content=****
SystemOut     O  added StringRefAddr: type=ws.isolated.class.loader, content=false
SystemOut     O  added StringRefAddr: type=mail.transport.protocol, content=smtp
SystemOut     O  added StringRefAddr: type=mail.imaps.class, content=com.sun.mail.imap.IMAPSSLStore
SystemOut     O  added StringRefAddr: type=mail.smtp.host, content=smtp.coldmail.com
SystemOut     O  added StringRefAddr: type=mail.debug, content=true
SystemOut     O  added StringRefAddr: type=mail.pop3s.class, content=com.sun.mail.pop3.POP3SSLStore
SystemOut     O  added StringRefAddr: type=mail.from, content=smith@coldmail.com
SystemOut     O  added StringRefAddr: type=mail.smtp.class, content=com.sun.mail.smtp.SMTPTransport
SystemOut     O  added StringRefAddr: type=mail.smtps.class, content=com.sun.mail.smtp.SMTPSSLTransport
SystemOut     O  added StringRefAddr: type=mail.imap.class, content=com.sun.mail.imap.IMAPStore
SystemOut     O  added StringRefAddr: type=mail.smtp.user, content=smith
SystemOut     O  added StringRefAddr: type=mail.pop3.class, content=com.sun.mail.pop3.POP3Store
SystemOut     O  added StringRefAddr: type=mail.mime.address.strict, content=true


SystemOut     O DEBUG: JavaMail version 1.4ea
SystemOut     O DEBUG: java.io.FileNotFoundException:
C:\Program Files\IBM\WebSphere\AppServer\java\jre\lib\javamail.providers
(The system cannot find the file specified.)
SystemOut     O DEBUG: !anyLoaded
SystemOut     O DEBUG: not loading resource: /META-INF/javamail.providers
SystemOut     O DEBUG: successfully loaded resource: /META-INF/javamail.default.providers
SystemOut     O DEBUG: Tables of loaded providers
SystemOut     O DEBUG: Providers Listed By Class Name:
     {com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider
[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Sun Microsystems, Inc],
com.sun.mail.smtp.SMTPTransport=javax.mail.Provider
[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun Microsystems, Inc],
com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider
[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun Microsystems, Inc],
com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider
[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun Microsystems, Inc],
com.sun.mail.imap.IMAPStore=javax.mail.Provider
[STORE,imap,com.sun.mail.imap.IMAPStore,Sun Microsystems, Inc],
com.sun.mail.pop3.POP3Store=javax.mail.Provider
[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun Microsystems, Inc]}
SystemOut     O DEBUG: Providers Listed By Protocol:
{imaps=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun Microsystems,Inc],
imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun Microsystems, Inc],
smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Sun Microsystems,Inc],
pop3=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun Microsystems, Inc],
pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun Microsystems, Inc],
smtp=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun Microsystems, Inc]}
SystemOut     O DEBUG: successfully loaded resource: /META-INF/javamail.default.address.map
SystemOut     O DEBUG: !anyLoaded
SystemOut     O DEBUG: not loading resource: /META-INF/javamail.address.map
SystemOut     O DEBUG: java.io.FileNotFoundException:
     C:\Program Files\IBM\WebSphere\AppServer\java\jre\lib\javamail.address.map
     (The system cannot find the file specified.)
SystemOut     O *** In SessionFactory.setPasswordAuthentication,
   TRANSPORT PasswordAuthentication is based on:
SystemOut     O  url=smtp://smith@smtp.coldmail.com
SystemOut     O  user=smith
SystemOut     O  password=****
SystemOut     O *** In SessionFactory.getObjectInstance, session properties:
SystemOut     O  mail.transport.protocol=smtp
SystemOut     O  mail.imaps.class=com.sun.mail.imap.IMAPSSLStore
SystemOut     O  mail.smtp.host=smtp.coldmail.com
SystemOut     O  mail.debug=true

SystemOut     O  mail.pop3s.class=com.sun.mail.pop3.POP3SSLStore
SystemOut     O  mail.from=smith@coldmail.com
SystemOut     O  mail.smtp.class=com.sun.mail.smtp.SMTPTransport
SystemOut     O  mail.smtps.class=com.sun.mail.smtp.SMTPSSLTransport
SystemOut     O  mail.imap.class=com.sun.mail.imap.IMAPStore
SystemOut     O  mail.smtp.user=smith
SystemOut     O  mail.pop3.class=com.sun.mail.pop3.POP3Store
SystemOut     O  mail.mime.address.strict=true
SystemOut     O DEBUG: mail.smtp.class property exists and points to com.sun.mail.smtp.SMTPTransport
SystemOut     O DEBUG SMTP: useEhlo true, useAuth false
SystemOut     O DEBUG SMTP: trying to connect to host "smtp.coldmail.com", port 25, isSSL false


javax.mail.MessagingException: Unknown SMTP host: smtp.coldmail.com;
  nested exception is:
 java.net.UnknownHostException: smtp.coldmail.com
 at com.sun.mail.smtp.SMTPTransport.openServer(SMTPTransport.java:1280)
 at com.sun.mail.smtp.SMTPTransport.protocolConnect(SMTPTransport.java:370)
 at javax.mail.Service.connect(Service.java:275)
 at javax.mail.Service.connect(Service.java:156)
 at javax.mail.Service.connect(Service.java:105)
 at javax.mail.Transport.send0(Transport.java:168)
 at javax.mail.Transport.send(Transport.java:98)
 at com.ibm.ws.mail.ut.TestServlet.doTask(TestServlet.java:104)
 at com.ibm.ws.mail.ut.TestServlet.doGet(TestServlet.java:65)
 at javax.servlet.http.HttpServlet.service(HttpServlet.java:707)
 at javax.servlet.http.HttpServlet.service(HttpServlet.java:820)
 at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1397)
 at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:759)
 at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:429)
 at com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest(ServletWrapperImpl.java:175)
 at com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:3512)
 at com.ibm.ws.webcontainer.webapp.WebGroup.handleRequest(WebGroup.java:273)
 at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:896)
 at com.ibm.ws.webcontainer.WSWebContainer.handleRequest(WSWebContainer.java:1530)
 at com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.java:161)
 at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimination(HttpInboundLink.java:455)
 at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleNewInformation(HttpInboundLink.java:384)
 at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.ready(HttpInboundLink.java:272)
 at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.sendToDiscriminators(NewConnectionInitialReadCallback.java:214)
 at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.complete(NewConnectionInitialReadCallback.java:113)
 at com.ibm.ws.tcp.channel.impl.AioReadCompletionListener.futureCompleted(AioReadCompletionListener.java:165)
 at com.ibm.io.async.AbstractAsyncFuture.invokeCallback(AbstractAsyncFuture.java:217)
 at com.ibm.io.async.AsyncChannelFuture.fireCompletionActions(AsyncChannelFuture.java:161)
 at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:138)
 at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:202)
 at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.java:766)
 at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:896)
 at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1487)
Caused by: java.net.UnknownHostException: smtp.coldmail.com
 at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:196)
 at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:366)
 at java.net.Socket.connect(Socket.java:519)
 at java.net.Socket.connect(Socket.java:469)
 at com.sun.mail.util.SocketFetcher.createSocket(SocketFetcher.java:232)
 at com.sun.mail.util.SocketFetcher.getSocket(SocketFetcher.java:189)
 at com.sun.mail.smtp.SMTPTransport.openServer(SMTPTransport.java:1250)
 ... 32 more
```

This output illustrates a connection failure to a Simple Mail Transfer Protocol (SMTP) server because a fictitious name, smtp.coldmail.com, is specified as the server name.

The following list provides tips on reading the previous sample of debugger output:

- The lines headed by `DEBUG` are printed by the mail provider at run time, while the two lines headed by `***` are printed by the application server at run time.
- In the second paragraph of code, the first few lines state that some configuration files are skipped. The mail component attempts to load a number of configuration files from different locations at run time. All

those files are not required. If a required file cannot be accessed, however, the mail component creates an exception. In this sample, there is no exception and the third-line announces that default providers are loaded.

- The next few lines, headed by either `Providers Listed by Class Name` or `Providers Listed by Protocols`, show the protocol providers that are loaded. The six providers that are listed are the default protocol providers that come under the built-in mail provider for the application server. If you install special service providers, and these providers are used in the current mail session, those providers will be listed here with the default providers.
- The two lines headed by `***` and the few lines below them are printed by the application server to show the configuration properties of the current mail session. Although these properties are listed by their internal name rather than the name you establish in the administrative console, you can easily recognize the relationships between them. For example, the mail.store.protocol property corresponds to the **Protocol** property in the **Incoming Mail Properties** section of the console panel for mail session configuration. Review the listed properties and values to verify that they correspond.
- The few lines above the exception stack show the mail activities when sending a message. First, the JavaMail API recognizes that the transport protocol is set to SMTP and that the com.sun.mail.smtp.SMTPTransport provider exists. Next, the output log displays the useEhlo and useAuth parameters, which are used by SMTP. Finally, the log shows the SMTP provider trying to connect to the smtp.coldmail.com mail server.
- The output log show the exception stack next. This data indicates that the specified mail server either does not exist or is not functioning.

Chapter 7. Security

Troubleshooting security configurations

The following topics help to troubleshoot specific problems that are related to configuring and enabling security configurations.

About this task

Refer to Security components troubleshooting tips for instructions on how to troubleshoot errors that are related to security.

Refer to SPNEGO TAI troubleshooting tips for instructions on how to troubleshoot errors that are related to diagnosing Simple and Protected GSS-API Negotiation (SPNEGO) trust association interceptor (TAI) problems and exceptions.

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO Web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

- Errors when configuring or enabling security
- Errors after enabling security
- Access problems after enabling security
- Errors after configuring or enabling Secure Sockets Layer
- Single sign-on configuration troubleshooting tips
- Enterprise Identity Mapping troubleshooting tips
- Authorization provider troubleshooting tips
- Password decoding troubleshooting tips

Security components troubleshooting tips

This document explains basic resources and steps for diagnosing security-related issues in WebSphere Application Server.

Basic resources and steps for diagnosing security-related issues in WebSphere Application Server include:

- What “Log files” on page 88 to look at and what to look for in them.
- What to look at and what to look for “Using SDSF” on page 89.
- “General approach for troubleshooting security-related issues” on page 90 to isolating and resolving security problems.
- When and how to “Trace security” on page 94.
- An overview and table of “CSIv2 CORBA minor codes” on page 95.

The following security-related problems are addressed elsewhere in the information center:

- Errors and access problems after enabling security

  After enabling security, a degradation in performance is realized. For more information about using unrestricted policy files, see the Enabling security for the realm section of the *Securing applications and their environment* PDF book.
- Errors after enabling SSL, or SSL-related error messages
- Errors trying to configure and enable security

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

For an overview of WebSphere Application Server security components such as Secure Authentication Services (SAS) and how they work in a distributed or an iSeries® environment, refer to the *Securing applications and their environment* PDF book.

Note: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Log files

When troubleshooting the security component, browse the Java Virtual Machine (JVM) logs for the server that hosts the resource you are trying to access. The following is a sample of messages you would expect to see from a server in which the security service has started successfully:

```
SASRas        A CWWSA0001I: Security configuration initialized.
SASRas        A CWWSA0002I: Authentication protocol: CSIV2/IBM
SASRas        A CWWSA0003I: Authentication mechanism: SWAM
SASRas        A CWWSA0004I: Principal name: MYHOSTNAME/aServerID
SASRas        A CWWSA0005I: SecurityCurrent registered.
SASRas        A CWWSA0006I: Security connection interceptor initialized.
SASRas        A CWWSA0007I: Client request interceptor registered.
SASRas        A CWWSA0008I: Server request interceptor registered.
SASRas        A CWWSA0009I: IOR interceptor registered.
NameServerImp I CWNMS0720I: Do Security service listener registration.
SecurityCompo A CWSCJ0242A: Security service is starting
UserRegistryI A CWSCJ0136I: Custom Registry:com.ibm.ws.security.registry.nt.
NTLocalDomainRegistryImpl has been initialized
SecurityCompo A CWSCJ0202A: Admin application initialized successfully
SecurityCompo A CWSCJ0203A: Naming application initialized successfully
SecurityCompo A CWSCJ0204A: Rolebased authorizer initialized successfully
SecurityCompo A CWSCJ0205A: Security Admin mBean registered successfully
SecurityCompo A CWSCJ0243A: Security service started successfully
SecurityCompo A CWSCJ0210A: Security enabled true
```

The following is an example of messages from a server which cannot start the security service, in this case because the administrative user ID and password given to communicate with the user registry is wrong, or the user registry itself is down or misconfigured:

```
SASRas        A CWWSA0001I: Security configuration initialized.
SASRas        A CWWSA0002I: Authentication protocol: CSIV2/IBM
SASRas        A CWWSA0003I: Authentication mechanism: SWAM
SASRas        A CWWSA0004I: Principal name: MYHOSTNAME/aServerID
SASRas        A CWWSA0005I: SecurityCurrent registered.
SASRas        A CWWSA0006I: Security connection interceptor initialized.
SASRas        A CWWSA0007I: Client request interceptor registered.
SASRas        A CWWSA0008I: Server request interceptor registered.
SASRas        A CWWSA0009I: IOR interceptor registered.
NameServerImp I CWNMS0720I: Do Security service listener registration.

SecurityCompo A CWSCJ0242A: Security service is starting
UserRegistryI A CWSCJ0136I: Custom Registry:com.ibm.ws.security.
registry.nt.NTLocalDomainRegistryImpl has been initialized
Authenticatio E CWSCJ4001E: Login failed for badID/<null>
javax.security.auth.login.LoginException: authentication failed: bad user/password
```

The following is an example of messages from a server for which Lightweight Directory Access Protocol (LDAP) has been specified as the security mechanism, but the LDAP keys have not been properly configured:

```
SASRas        A CWWSA0001I: Security configuration initialized.
SASRas        A CWWSA0002I: Authentication protocol: CSIV2/IBM
SASRas        A CWWSA0003I: Authentication mechanism: LTPA
SASRas        A CWWSA0004I: Principal name: MYHOSTNAME/anID
SASRas        A CWWSA0005I: SecurityCurrent registered.
SASRas        A CWWSA0006I: Security connection interceptor initialized.
SASRas        A CWWSA0007I: Client request interceptor registered.
SASRas        A CWWSA0008I: Server request interceptor registered.
SASRas        A CWWSA0009I: IOR interceptor registered.
NameServerImp I CWNMS0720I: Do Security service listener registration.
SecurityCompo A CWSCJ0242A: Security service is starting
UserRegistryI A CWSCJ0136I: Custom Registry:com.ibm.ws.security.registry.nt.
NTLocalDomainRegistryImpl has been initialized
SecurityServe E CWSCJ0237E: One or more vital LTPAServerObject configuration
attributes are null or not available. The attributes and values are password :
LTPA password does exist, expiration time 30, private key <null>, public key <null>,
and shared key <null>.
```

A problem with the Secure Sockets Layer (SSL) configuration might lead to the following message. Ensure that the keystore location and keystore passwords are valid. Also, ensure the keystore has a valid personal certificate and that the personal certificate public key or certificate authority (CA) root has been extracted on put into the truststore.

```
SASRas        A CWWSA0001I: Security configuration initialized.
SASRas        A CWWSA0002I: Authentication protocol: CSIV2/IBM
SASRas        A CWWSA0003I: Authentication mechanism: SWAM
SASRas        A CWWSA0004I: Principal name: MYHOSTNAME/aServerId
SASRas        A CWWSA0005I: SecurityCurrent registered.
SASRas        A CWWSA0006I: Security connection interceptor initialized.
SASRas        A CWWSA0007I: Client request interceptor registered.
SASRas        A CWWSA0008I: Server request interceptor registered.
SASRas        A CWWSA0009I: IOR interceptor registered.
SASRas        E CWWSA0026E: [SecurityTaggedComponentAssistorImpl.register]
Exception connecting object to the ORB.  Check the SSL configuration to ensure
 that the SSL keyStore and trustStore properties are set properly.  If the problem
persists, contact support for assistance. org.omg.CORBA.OBJ_ADAPTER:
ORB_CONNECT_ERROR (5) - couldn't get Server Subcontract  minor code:
4942FB8F  completed: No
```

Using SDSF

When troubleshooting the security component, use System Display and Search Facility (SDSF) to browse logs for the server that hosts the resource you are trying to access. The following sample of messages helps you see from a server in which the security service has started successfully:

```
 +BBOM0001I com_ibm_authMechanisms_type_OID: No OID for this mechanism.
 +BBOM0001I com_ibm_security_SAF_unauthenticated: WSGUEST.
 +BBOM0001I com_ibm_security_SAF_EJBROLE_Audit_Messages_Suppress: 0.
 +BBOM0001I com_ibm_ws_logging_zos_errorlog_format_cbe: NOT SET, 280
 DEFAULT=0.
 +BBOM0001I com_ibm_CSI_performClientAuthenticationRequired: 0.
 +BBOM0001I com_ibm_CSI_performClientAuthenticationSupported: 1.
 +BBOM0001I com_ibm_CSI_performTransportAssocSSLTLSRequired: 0.
 +BBOM0001I com_ibm_CSI_performTransportAssocSSLTLSSupported: 1.
 +BBOM0001I com_ibm_CSI_rmiInboundPropagationEnabled: 1.
 +BBOM0001I com_ibm_CSI_rmiOutboundLoginEnabled: 0.
 +BBOM0001I com_ibm_CSI_rmiOutboundPropagationEnabled: 1.
 +BBOM0001I security_assertedID_IBM_accepted: 0.
 +BBOM0001I security_assertedID_IBM_sent: 0.
```

```
+BBOM0001I security_disable_daemon_ssl: NOT SET, DEFAULT=0.
+BBOM0001I security_sslClientCerts_allowed: 0.
+BBOM0001I security_sslKeyring: NOT SET.
+BBOM0001I security_zOS_domainName: NOT SET.
+BBOM0001I security_zOS_domainType: 0.
+BBOM0001I security_zSAS_ssl_repertoire: SY1/DefaultIIOPSSL.
+BBOM0001I security_EnableRunAsIdentity: 0.
+BBOM0001I security_EnableSyncToOSThread: 0.
+BBOM0001I server_configured_system_name: SY1.
+BBOM0001I server_generic_short_name: BBOC001.
+BBOM0001I server_generic_uuid:  457
*** Message beginning with BBOO0222I apply to Java within ***
*** WebSphere Application Server Security ***
+BBOO0222I: SECJ6004I: Security Auditing is disabled.
+BBOO0222I: SECJ0215I: Successfully set JAAS login provider 631
configuration class to com.ibm.ws.security.auth.login.Configuration.
+BBOO0222I: SECJ0136I: Custom 632
Registry:com.ibm.ws.security.registry.zOS.SAFRegistryImpl has been initialized
+BBOO0222I: SECJ0157I: Loaded Vendor AuthorizationTable: 633
com.ibm.ws.security.core.SAFAuthorizationTableImpl
```

General approach for troubleshooting security-related issues

When troubleshooting security-related problems, the following questions are very helpful:

Does the problem occur when security is disabled?

This question is a good litmus test to determine that a problem is security related. However, just because a problem only occurs when security is enabled does not always make it a security problem. More troubleshooting is necessary to ensure the problem is really security-related.

Did security seem to initialize properly?

A lot of security code is visited during initialization. So you can see problems there first if the problem is configuration related.

The following sequence of messages that are generated in the `SystemOut.log` indicate normal code initialization of an application server. This sequence varies based on the configuration, but the messages are similar:

```
SASRas        A CWWSA0001I: Security configuration initialized.
SASRas        A CWWSA0002I: Authentication protocol: CSIV2/IBM
SASRas        A CWWSA0003I: Authentication mechanism: SWAM
SASRas        A CWWSA0004I: Principal name: BIRKT20/pbirk
SASRas        A CWWSA0005I: SecurityCurrent registered.
SASRas        A CWWSA0006I: Security connection interceptor initialized.
SASRas        A CWWSA0007I: Client request interceptor registered.
SASRas        A CWWSA0008I: Server request interceptor registered.
SASRas        A CWWSA0009I: IOR interceptor registered.
NameServerImp I CWNMS0720I: Do Security service listener registration.
SecurityCompo A CWSCJ0242A: Security service is starting
UserRegistryI A CWSCJ0136I: Custom Registry:com.ibm.ws.security.registry.nt.
NTLocalDomainRegistryImpl has been initialized
SecurityCompo A CWSCJ0202A: Admin application initialized successfully
SecurityCompo A CWSCJ0203A: Naming application initialized successfully
SecurityCompo A CWSCJ0204A: Rolebased authorizer initialized successfully
SecurityCompo A CWSCJ0205A: Security Admin mBean registered successfully
SecurityCompo A CWSCJ0243A: Security service started successfully

SecurityCompo A CWSCJ0210A: Security enabled true
```

The following sequence of messages generated in the SDSF active log indicate normal code initialization of an application server. Non-security messages have been removed from the sequence that follows. This sequence will vary based on the configuration, but the messages are similar:

```
Trace: 2005/05/06 17:27:31.539 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: printProperties
  SourceId: com.ibm.ws390.orb.CommonBridge
  Category: AUDIT
  ExtendedMessage: BBOJ0077I java.security.policy =
         /WebSphere/V6R1M0/AppServer/profiles/default/pr
Trace: 2005/05/06 17:27:31.779 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: printProperties
  SourceId: com.ibm.ws390.orb.CommonBridge
  Category: AUDIT
  ExtendedMessage: BBOJ0077I java.security.auth.login.config =
         /WebSphere/V6R1M0/AppServer/profiles/default/pr
Trace: 2005/05/06 17:27:40.892 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.core.SecurityDM
  SourceId: com.ibm.ws.security.core.SecurityDM
  Category: INFO
  ExtendedMessage: BBOO0222I: SECJ0231I: The Security component's FFDC
         Diagnostic Module com.ibm.ws.security.core.Secur
red successfully: true.
Trace: 2005/05/06 17:27:40.892 01 t=8E96E0 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 932
  error message: BBOO0222I: SECJ0231I: The Security component's FFDC
         Diagnostic Module com.ibm.ws.security.core.Securit
d successfully: true.
Trace: 2005/05/06 17:27:41.054 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.audit.AuditServiceImpl
  SourceId: com.ibm.ws.security.audit.AuditServiceImpl
  Category: AUDIT
  ExtendedMessage: BBOO0222I: SECJ6004I: Security Auditing is disabled.
Trace: 2005/05/06 17:27:41.282 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.core.distSecurityComponentImpl
  SourceId: com.ibm.ws.security.core.distSecurityComponentImpl
  Category: INFO
  ExtendedMessage: BBOO0222I: SECJ0309I: Java 2 Security is disabled.
Trace: 2005/05/06 17:27:41.282 01 t=8E96E0 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 932
  error message: BBOO0222I: SECJ0309I: Java 2 Security is disabled.
Trace: 2005/05/06 17:27:42.239 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.auth.login.Configuration
  SourceId: com.ibm.ws.security.auth.login.Configuration
  Category: AUDIT
  ExtendedMessage: BBOO0222I: SECJ0215I: Successfully set JAAS login
         provider configuration class to com.ibm.ws.securit
Configuration.
Trace: 2005/05/06 17:27:42.253 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.core.distSecurityComponentImpl
  SourceId: com.ibm.ws.security.core.distSecurityComponentImpl
  Category: INFO
  ExtendedMessage: BBOO0222I: SECJ0212I: WCCM JAAS configuration information
         successfully pushed to login provider clas
Trace: 2005/05/06 17:27:42.254 01 t=8E96E0 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 932
  error message: BBOO0222I: SECJ0212I: WCCM JAAS configuration information
          successfully pushed to login provider class.
Trace: 2005/05/06 17:27:42.306 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.core.distSecurityComponentImpl
  SourceId: com.ibm.ws.security.core.distSecurityComponentImpl
  Category: INFO
  ExtendedMessage: BBOO0222I: SECJ0240I: Security service initialization
         completed successfully
Trace: 2005/05/06 17:27:42.306 01 t=8E96E0 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 932
  error message: BBOO0222I: SECJ0240I: Security service initialization
          completed successfully
Trace: 2005/05/06 17:27:42.952 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.objectpool.ObjectPoolService
  SourceId: com.ibm.ws.objectpool.ObjectPoolService
  Category: INFO
  ExtendedMessage: BBOO0222I: OBPL0007I: Object Pool Manager service
         is disabled.
Trace: 2005/05/06 17:27:53.512 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.registry.UserRegistryImpl
  SourceId: com.ibm.ws.security.registry.UserRegistryImpl
  Category: AUDIT
  ExtendedMessage: BBOO0222I: SECJ0136I: Custom
         Registry:com.ibm.ws.security.registry.zOS.SAFRegistryImpl
         has been init
Trace: 2005/05/06 17:27:55.229 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.role.PluggableAuthorizationTableProxy
  SourceId: com.ibm.ws.security.role.PluggableAuthorizationTableProxy
  Category: AUDIT
  ExtendedMessage: BBOO0222I: SECJ0157I: Loaded Vendor
         AuthorizationTable: com.ibm.ws.security.core.SAFAuthorizationTab
Trace: 2005/05/06 17:27:56.481 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.core.distSecurityComponentImpl
  SourceId: com.ibm.ws.security.core.distSecurityComponentImpl
  Category: INFO
  ExtendedMessage: BBOO0222I: SECJ0243I: Security service started successfully
Trace: 2005/05/06 17:27:56.481 01 t=8E96E0 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 932
  error message: BBOO0222I: SECJ0243I: Security service started successfully
Trace: 2005/05/06 17:27:56.482 01 t=8E96E0 c=UNK key=P8 (13007002)
  ThreadId: 0000000a
  FunctionName: com.ibm.ws.security.core.distSecurityComponentImpl
  SourceId: com.ibm.ws.security.core.distSecurityComponentImpl
  Category: INFO
  ExtendedMessage: BBOO0222I: SECJ0210I: Security enabled true
Trace: 2005/05/06 17:27:56.483 01 t=8E96E0 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 932
  error message: BBOO0222I: SECJ0210I: Security enabled true
```

Is there a stack trace or exception printed in the system log file?

A single stack trace tells a lot about the problem. What code initiated the code that failed? What is the failing component? Which class did the failure actually come from? Sometimes the stack trace is all that is needed to solve the problem and it can pinpoint the root cause. Other times, it can only give us a clue, and can actually be misleading. When support analyzes a stack trace, they can request additional trace if it is not clear what the problem is. If it seems to be security-related and the solution cannot be determined from the stack trace or problem description, you are asked to gather the following trace specification: SASRas=all=enabled:com.ibm.ws.security.*=all=enabled from all processes involved.

Is this a distributed security problem or a local security problem?

- If the problem is local, that is the code involved does not make a remote method invocation, then troubleshooting is isolated to a single process. It is important to know when a problem is local versus distributed because the behavior of the object request broker (ORB), among other components, is different between the two. When a remote method invocation takes place, an entirely different security code path is entered.
- When you know that the problem involves two or more servers, the techniques of troubleshooting change. You need to trace all the servers involved simultaneously so that the trace shows the client and server sides of the problem. Make sure the timestamps on all machines match as closely as possible so that you can find the request and reply pair from two different processes. Enable both Secure Authentication Services (SAS) or z/SAS and Security trace using the trace specification: SASRas=all=enabled:com.ibm.ws.security.*=all=enabled.

  For more information on enabling trace, see Enabling trace.

For more information on enabling trace, see Working with Trace.

Is the problem related to authentication or authorization?

Most security problems fall under one of these two categories. Authentication is the process of determining who the caller is. Authorization is the process of validating that the caller has the proper authority to invoke the requested method. When authentication fails, typically this failure is related to either the authentication protocol, authentication mechanism or user registry. When authorization fails, this is usually related to the application bindings from assembly and deployment and to the caller's identity who is accessing the method and the roles that are required by the method.

Is this a Web or EJB request?

Web requests have a completely different code path than Enterprise JavaBeans (EJB) requests. Different security features exist for Web requests than for EJB requests, requiring a completely different body of knowledge to resolve. For example, when using the Lightweight Third-Party Authentication (LTPA) authentication mechanism, the single sign-on feature (SSO) is available for Web requests but not for EJB requests. Web requests involve HTTP header information that is not required by EJB requests due to the protocol differences. Also, the Web container or servlet engine is involved in the entire process. Any of these components can be involved in the problem and all require consideration during troubleshooting, based on the type of request and where the failure occurs.

Secure EJB requests heavily involve the ORB and Naming components since they flow over the RMI/IIOP protocol. In addition, when Workload Manager (WLM) is enabled, other behavior changes in the code can be observed. All of these components interact closely for security to work properly in this environment. At times, trace in any or all of these components might be necessary to troubleshoot problems in this area.

The trace specification to begin with is SASRas=all=enabled:com.ibm.ws.security.*=all=enabled. ORB trace is also very beneficial when the SAS/Security trace does not seem to pinpoint the problem.

Does the problem seem to be related to the Secure Sockets Layer (SSL)?

SSL is a totally distinct separate layer of security. Troubleshooting SSL problems is usually separate from troubleshooting authentication and authorization problems, and you have many considerations. Usually, SSL problems are first-time setup problems because the configuration can be difficult. Each client must contain the signer certificate of the server. During mutual authentication, each server must contain the client's signer certificate. Also, there can be protocol differences (SSLv3 vs. Transport Layer Security (TLS)), and listener port problems related to stale Interoperable Object References (IORs), that is IORs from a server, that reflect the port prior to the server restarting.

For SSL problems, sometimes you get a request for an SSL trace to determine what is happening with the SSL handshake. The SSL handshake is the process that occurs when a client opens a socket to a server. If anything goes wrong with the key exchange, cipher exchange, and so on, the handshake fails and the socket is not valid. Tracing JSSE (the SSL implementation that is used in WebSphere Application Server) involves the following steps:

- Set the following system property on the client and server processes: -Djavax.net.debug=true. For the server, add the system property to the generic JVM arguments property of the JVM settings page. For more information on this task, refer to Java virtual machine settings section of the *Administering applications and their environment* PDF book.
- Turn on ORB trace as well.
- Recreate the problem.

  The `SystemOut.log` of both processes contain the JSSE trace. You can find trace similar to the following example:

```
SSLConnection: install <com.ibm.sslite.e@3ae78375>
>> handleHandshakeV2 <com.ibm.sslite.e@3ae78375>
>> handshakeV2 type = 1
>> clientHello: SSLv2.
SSL client version: 3.0
...
```

```
...
...
JSSEContext: handleSession[Socket[addr=null,port=0,localport=0]]

<< sendServerHello.
SSL version: 3.0
SSL_RSA_WITH_RC4_128_MD5
HelloRandom
...
...
...
<< sendCertificate.
<< sendServerHelloDone.
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleHandshake <com.ibm.sslite.e@3ae78375>
>> handshakeV3 type = 16

>> clientKeyExchange.
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleChangeCipherSpec <com.ibm.sslite.e@3ae78375>
>> handleData <com.ibm.sslite.e@3ae78375>
>> handleHandshake <com.ibm.sslite.e@3ae78375>
>> handshakeV3 type = 20
>> finished.
<< sendChangeCipherSpec.
<< sendFinished.
```

Trace security

The classes that implement WebSphere Application Server security are:

- com.ibm.ws.security.*
- com.ibm.websphere.security.*
- com.ibm.WebSphereSecurityImpl.*
- SASRas
- com.ibm.ws.wim.* for tracing with a Virtual Member Manager (VMM) repository

To view detailed information on the run time behavior of security, enable trace on the following components and review the output:

- com.ibm.ws.security.*=all=enabled:com.ibm.WebSphereSecurityImpl.*= all=enabled:com.ibm.websphere.security.*=all=enabled. This trace statement collects the trace for the security runtime.
- com.ibm.ws.console.security.*=all=enabled. This trace statement collects the trace for the security center administrative console.
- SASRas=all=enabled. This trace statement collects the trace for SAS (low-level authentication logic).
- com.ibm.ws.wim.*=all=enabled:com.ibm.websphere.wim.*=all=enabled. This trace statement collects the trace for VMM.

Fine tuning Security traces:

If a subset of packages need to be traced, specify a trace specification more detailed than com.ibm.ws.security.*=all=enabled. For example, to trace just dynamic policy code, you can specify com.ibm.ws.security.policy.*=all=enabled. To disable dynamic policy trace, you can specify com.ibm.ws.security.policy.*=all=disabled.

Configuring CSIv2, or SAS Trace Settings

Situations arise where reviewing trace for the CSIv2 or SAS authentication protocols can assist in troubleshooting difficult problems. This section describes how to enable to CSIv2 and SAS trace.

Enabling Client-Side CSIv2 and SAS Trace

To enable CSIv2 and SAS trace on a pure client, the following steps need to be taken:

- Edit the file `TraceSettings.properties` in the `/WebSphere/AppServer/properties` directory. For example, edit *profile_root*`/properties/TraceSettings.properties`.
- In this file, change traceFileName= to point to the path in which you want the output file created. For example, traceFileName=*profile_root*`/logs/sas_client`.
- n this file, add the trace specification string: SASRas=all=enabled. Any additional trace strings can be added on separate lines.

- Point to this file from within your client application. On the Java command line where you launch the client, add the following system property: -DtraceSettingsFile=TraceSettings.properties.

  Note: Do not give the fully qualified path to the TraceSettings.properties file. Make sure that the TraceSettings.properties file is in your class path.

Enabling Server-Side CSIv2 and SAS Trace

To enable SAS trace in an application server, complete the following:

- Add the trace specification, SASRas=all=enabled, to the server.xml file or add it to the Trace settings within the WebConsole GUI.
- Typically it is best to also trace the authorization security runtime in addition to the authentication protocol runtime. To do this, use the following two trace specifications in combination: SASRas=all=enabled:com.ibm.ws.security.*=all=enabled.
- When troubleshooting a connection type problem, it is beneficial to trace both CSIv2 and SAS or CSIv2 and z/SAS and the ORB. To do this, use the following three trace specifications: SASRas=all=enabled:com.ibm.ws.security.*=all=enabled:ORBRas=all=enabled.
- In addition to adding these trace specifications, for ORB trace there are a couple of system properties that also need to be set. Go to the ORB settings in the GUI and add the following two properties: com.ibm.CORBA.Debug=true and com.ibm.CORBA.CommTrace=true.

CSIv2 CORBA minor codes

Whenever exceptions occur within the security code on either the client or server, the eventual exception becomes a Common Object Request Broker Architecture (CORBA) exception. Any exception that occurs gets embedded in a CORBA exception because the CORBA architecture is used by the security service for its own inter-process communication. CORBA exceptions are generic and indicate a problem in communication between two components. CORBA minor codes are more specific and indicate the underlying reason that a component could not complete a request.

The following shows the CORBA minor codes that a client can expect to receive after running a security-related request such as authentication. It also includes the CORBA exception type that the minor code appears in.

The following exception shows an example of a CORBA exception where the minor code is 49424300 and indicates Authentication Failure. Typically, a descriptive message is also included in the exception to assist in troubleshooting the problem. Here, the detailed message is: ″Exception caught invoking authenticateBasicAuthData from SecurityServer for user jdoe. Reason: com.ibm.WebSphereSecurity.AuthenticationFailedException″ which indicates that the authentication failed for user jdoe.

The completed field in the exception indicates whether the method was completed or not. In the case of a NO_PERMISSION, never invoke the message; therefore it is always completed:No. Other exceptions that are caught on the server side can have a completed status of ″Maybe″ or ″Yes″.

```
org.omg.CORBA.NO_PERMISSION: Caught WSSecurityContextException in
WSSecurityContext.acceptSecContext(),
reason: Major Code[0] Minor Code[0] Message[Exception caught invoking
authenticateBasicAuthData from SecurityServer for user jdoe.  Reason:
com.ibm.WebSphereSecurity.AuthenticationFailedException]  minor code: 49424300
completed: No

at com.ibm.ISecurityLocalObjectBaseL13Impl.PrincipalAuthFailReason.
map_auth_fail_to_minor_code(PrincipalAuthFailReason.java:83)
        at com.ibm.ISecurityLocalObjectBaseL13Impl.CSIServerRI.receive_request
               (CSIServerRI.java:1569)
        at com.ibm.rmi.pi.InterceptorManager.iterateReceiveRequest
        (InterceptorManager.java:739)
    at com.ibm.CORBA.iiop.ServerDelegate.dispatch(ServerDelegate.java:398)
    at com.ibm.rmi.iiop.ORB.process(ORB.java:313)
    at com.ibm.CORBA.iiop.ORB.process(ORB.java:1581)
    at com.ibm.rmi.iiop.GIOPConnection.doWork(GIOPConnection.java:1827)
    at com.ibm.rmi.iiop.WorkUnitImpl.doWork(WorkUnitImpl.java:81)
    at com.ibm.ejs.oa.pool.PooledThread.run(ThreadPool.java:91)
    at com.ibm.ws.util.CachedThread.run(ThreadPool.java:149)
```

The following table shows the CORBA minor codes which a client can expect to receive after running a security-related request such as authentication. It also includes the CORBA exception type that the minor code would appear in.

Table 1.

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA .*)	Minor code description	Retry performed (when authenticationRetryEnabled = true)
AuthenticationFailed	49424300	NO_PERMISSION	This code is a generic authentication failed error. It does not give any details about whether or not the user ID or password is valid. Some user registries can choose to use this type of error code, others can choose to use the next three types that are more specific.	Yes
InterceptLocateException	494210B8	INTERNAL	This indicates a problem when processing an incoming locate request.	No
InvalidUserid	49424301	NO_PERMISSION	This code occurs when the registry returns bad user ID.	Yes
InvalidPassword	49424302	NO_PERMISSION	This code occurs when the registry returns a bad password.	Yes
InvalidSecurityCredentials	49424303	NO_PERMISSION	This is a generic error indicating that the credentials are bad for some reason. It might be that the right attributes are not set.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
InvalidRealm	49424304	NO_PERMISSION	This code occurs when the REALM in the token received from the client does not match the server's current realm.	No
ValidationFailed	49424305	NO_PERMISSION	A validation failure occurs when a token is sent from the client or server to a target server but the token format or the expiration is not valid.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
CredentialTokenExpired	49424306	NO_PERMISSION	This code is more specific about why the validation failed. In this case, the token has an absolute lifetime and the lifetime has expired. Therefore, it is no longer a valid token and cannot be used.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
InvalidCredentialToken	49424307	NO_PERMISSION	This is more specific about why the validation failed. In this case, the token cannot be decrypted or the data within the token is not readable.	Yes, if client has BasicAuth credential (token based credential was rejected in the first place).
SessionDoesNotExist	49424308	NO_PERMISSION	This indicates that the CSIv2 session does not exist on the server. Typically, a retry occurs automatically and successfully creates a new session.	Yes

Table 1. (continued)

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA .*)	Minor code description	Retry performed (when authenticationRetryEnabled = true)
SessionConflictingEvidence	49424309	NO_PERMISSION	This indicates that a session already exists on the server that matches the context_id sent over by the client. However, the information provided by the client for this EstablishContext message is different from the information originally provided to establish the session.	Yes
SessionRejected	4942430A	NO_PERMISSION	This indicates that the session referenced by the client has been previously rejected by the server.	Yes
SecurityServerNotAvailable	4942430B	NO_PERMISSION	This error occurs when the server cannot contact the local or remote security server in order to authenticate or validate.	No
InvalidIdentityToken	4942430C	NO_PERMISSION	This error indicates that identity cannot be obtained from the identity token when Identity Assertion is enabled.	No
IdentityServerNotTrusted	4942430D	NO_PERMISSION	This indicates that the server ID of the sending server is not on the target server's trusted principal list.	No
InvalidMessage	4942430E	NO_PERMISSION	This indicates that the CSIv2 message format is not valid for the receiving server.	No
AuthenticationNotSupported	49421090	NO_PERMISSION	This error occurs when a mechanism does not support authentication (very rare).	No
InvalidSecurityMechanism	49421091	NO_PERMISSION	This is used to indicate that the specified security mechanism is not known.	No
CredentialNotAvailable	49421092	NO_PERMISSION	This indicates a credential is not available when it is required.	No
SecurityMechanismNotSupported	49421093	NO_PERMISSION	This error occurs when a security mechanism that is specified in the CSIv2 token is not implemented on the server.	No
ValidationNotSupported	49421094	NO_PERMISSION	This error occurs when a mechanism does not support validation, such as LocalOS. This error does not occur since the LocalOS credential is not a forwardable credential, therefore, validation never needs to be called on this credential.	No
CredentialTokenNotSet	49421095	NO_PERMISSION	This is used to indicate that the token inside the credential is null.	No
ServerConnectionFailed	494210A0	COMM_FAILURE	This error is used when a connection attempt fails.	Yes (via ORB retry)
CorbaSystemException	494210B0	INTERNAL	This code is a generic CORBA specific exception in system code.	No
JavaException	494210B1	INTERNAL	This is a generic error that indicated that an unexpected Java exception occurred.	No

Table 1. (continued)

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA .*)	Minor code description	Retry performed (when authenticationRetryEnabled = true)
ValueIsNull	494210B2	INTERNAL	This code is used to indicate that a value or parameter that passed in is null.	No
EffectivePolicyNotPresent	494210B3	INTERNAL	This indicates that an effective policy object for CSIv2 is not present. This object is used to determine what security configuration features are specified.	No
NullPointerException	494210B4	INTERNAL	This code is used to indicate that a NullPointerException is caught in the runtime.	No
ErrorGettingClassInstance	494210B5	INTERNAL	This indicates a problem loading a class dynamically.	No
MalFormedParameters	494210B6	INTERNAL	This indicates parameters are not valid.	No
DuplicateSecurityAttributeType	494210B7	INTERNAL	This indicates a duplicate credential attribute that is specified during the set_attributes operation.	No
MethodNotImplemented	494210C0	NO_IMPLEMENT	This indicates that a method invoked is not implemented.	No
GSSFormatError	494210C5	BAD_PARAM	This code indicates that a Generic Security Services (GSS) encoding or decoding routine has created an exception.	No
TagComponentFormatError	494210C6	BAD_PARAM	This code indicates that a tag component cannot be read properly.	No
InvalidSecurityAttributeType	494210C7	BAD_PARAM	This code indicates an attribute type specified during the set_attributes operation is not a valid type.	No
SecurityConfigError	494210CA	INITIALIZE	This code indicates a problem exists between the client and server configuration.	No

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Security configuration and enablement errors

Use this information to troubleshoot problems with configuring or enabling security.

What kind of error are you seeing?

- “″LTPA password not set. validation failed″ message displayed as error in the administrative console after saving administrative or application security settings ” on page 99
- “WebSphere Application Server Version 6 is not working correctly with Enterprise Workload Manager (EWLM)” on page 99
- If you successfully configured security, but are now having problems accessing Web resources or the administrative console, refer to Errors or access problems after enabling security.
- “NMSV0610I: A NamingException is being thrown from a javax.naming.Context implementation” on page 99
- “When administrative security is enabled but application security is not enabled, the performance servlet displays authorization errors and cannot provide statistics” on page 100

- “″Name value is invalid″ displays when migrating users and groups after the JACC provider for Tivoli is configured” on page 100
- “A Sun JDK can not read a PKCS12 keystore created by the Application Server” on page 100

For general tips on diagnosing and resolving security-related problems, see the topic Troubleshooting the security component.

″LTPA password not set. validation failed″ message displayed as error in the administrative console after saving administrative or application security settings

This error can be caused if, when configuring WebSphere Application Server security, LTPA is selected as the authentication mechanism and the LTPA password field is not set. To resolve this problem:

- Select **Security > Global security > Authentication mechanisms and expiration** .
- Complete the password and confirm password fields.
- Click **OK**.
- Try setting administrative or application security again.

WebSphere Application Server Version 6 is not working correctly with Enterprise Workload Manager™ (EWLM)

To use WebSphere Application Server Version 6 with EWLM, you must manually update the WebSphere Application Server server.policy files. For example:

```
grant codeBase "file:/<EWLM_Install_Home>/classes/ARM/arm4.jar" {
 permission java.security.AllPermission;
};
```

Otherwise, you might encounter a Java 2 security exception for violating the Java 2 security permission.

For more information on configuring server.policy files, refer to the server.policy file permissions section in the *Developing and deploying applications* PDF book.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

NMSV0610I: A NamingException is being thrown from a javax.naming.Context implementation

If you use CSIv2 inbound authentication, basic authentication is required, and Java clients running with com.ibm.CORBA.validateBasicAuth=true might fail with the following exception:

```
If you use CSIv2 inbound authentication, basic authentication is required, and Java™
clients running with com.ibm.CORBA.validateBasicAuth=true might fail with the
following exception:

NMSV0610I: A NamingException is being thrown from a javax.naming.Context
           implementation. Details follow:

Context implementation: com.ibm.ws.naming.jndicos.CNContextImpl
Context method: lookupExt
Context name: TestaburgerNode01Cell/nodes/TestaburgerNode01/servers/server1
Target name: SecurityServer
Other data: ""
Exception stack trace: javax.naming.NoPermissionException: NO_PERMISSION
exception caught. Root exception is org.omg.CORBA.NO_PERMISSION:
vmcid: 0x49421000 minor code: 92 completed: No
...
```

```
SECJ0395E: Could not locate the SecurityServer at host/port:9.42.72.27/9100
to validate the userid and password entered. You may need to specify valid
securityServerHost/Port in (WAS_INSTALL_ROOT)/properties/sas.client.props file.
```

To fix this problem, modify the `com.ibm.CORBA.validateBasicAuth=false` property in the client's `sas.clients.props` file, which is located in WAS_HOME/profiles/<profile-name>/properties, and then run the client.

When administrative security is enabled but application security is not enabled, the performance servlet displays authorization errors and cannot provide statistics

In WebSphere Application Server Version 6.1, when administrative security is enabled, the administration service is locked down. However, if application security is not enabled, an authentication challenge does not occur for incoming requests and, consequently, credentials do not exist for the performance servlet to access the administration service.

If administrative security is enabled, you also must enable application security for the performance servlet to process incoming requests.

″Name value is invalid″ displays when migrating users and groups after the JACC provider for Tivoli® is configured

When you use the migrateEAR utility to migrate the changes that were made to console users and groups after the JACC provider for Tivoli Access Manager is configured, the following configuration error displays in the systemOut.log file.

```
<specialSubjects> name value is invalid
```

The migrateEAR utility migrates the user and group data that is contained in the admin-authz.xml file. However, the migrateEAR utility does not convert the XML tags that are listed in the admin-authz.xml file if the pdwas-admin group is added to the administrator access control list (ACL) in Tivoli Access Manager prior to migration.

To resolve this error, enter the following command in padadmin to check whether the pdwas-admin group is in the administrator ACL before you migrate:

```
acl show
_WebAppServer_deployedResources_Roles_administrator_admin-authz_ACL
```

The following result should display:

```
ACL Name:
_WebAppServer_deployedResources_Roles_administrator_admin-authz_ACL
Description: Created by the Tivoli Access Manager
for Websphere Application Server Migration Tool.
Entries:
User sec_master TcmdbsvaBR1
Group pdwas-admin T[WebAppServer]i
```

If the pdwas-admin group is not listed, then enter the following command in pdadmin to modify the ACL to add the pdwas-admin group:

```
acl modify
_WebAppServer_deployedResources_Roles_administrator_admin
-authz_ACL set gruop pdwas-admin T [WebAppServer]i
```

A Sun JDK can not read a PKCS12 keystore created by the Application Server

A Sun JDK is not able to read a PKCS12 keystore created by the Application Server. The reason for this is that the PKCS12 implementation used by the IBM SDK and the Application Server is different than the implementation used by the Sun JDK. The difference causes problems when a Sun JDK is used to read the default trustore, trust.p12, or keystore, key.P12 created by the Application Server.

Because the truststore can not be read by the Sun JDK, you must first extract the certificates from the trustore using an IBM SDK. You can then import these certificates into a keystore that the Sun JDK can recognize correctly, such as a JKS keystore.

Security enablement followed by errors

Use this information if you are experiencing errors after security is enabled.

What kind of error are you seeing?

- Authentication error accessing a Web page
- Authorization error accessing a Web page
- "Authentication fails when code pages differ between the client and the server" on page 102
- Error Message: `CWSCJ0314E: Current Java 2 security policy reported a potential violation`
- `CWMSG0508E: The JMS Server security service was unable to authenticate user ID:` error displayed in SystemOut.log when starting an application server
- Error Message: `CWSCJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available` after enabling security and starting the application server
- An AccessControlException is reported in the SystemOut.log
- Error Message: `CWSCJ0336E: Authentication failed for user {0} because of the following exception {1}`
- "Error Message: SECJ0352E: Could not get the users matching the pattern {0} because of the following exception {1}" on page 105
- "Generate keys error when using the Profile Management tool to create a new profile" on page 106

For general tips on diagnosing and resolving security-related problems, see the topic Troubleshooting the security component.

IBM Support has documents and tools that can save you time gathering information needed to resolve problems as described in Troubleshooting help from IBM. Before opening a problem report, see the Support page:

- http://www.ibm.com/servers/eserver/support/iseries/allproducts/index.html

Authentication error accessing a Web page

Possible causes for authentication errors include:

- **Incorrect user name or passwords.** Check the user name and password and make sure that they are correct.
- **Security configuration error : User registry type is not set correctly.** Check the user registry property in administrative security settings in the administrative console. Verify that the user registry property is the intended user registry.
- **Internal program error.** If the client application is a Java standalone program, this program might not gather or send credential information correctly.

If the user registry configuration, user ID, and password appear correct, use the WebSphere Application Server trace to determine the cause of the problem. To enable security trace, use the com.ibm.ws.security.*=all=enabled trace specification.

Authorization error accessing a Web page

If a user who is supposed to have access to a resource does not, a configuration step is probably missing. For more information on configuring access to resources, review the chapter Authorizing access to administrative roles in the *Securing applications and their environment* PDF book.

Specifically:

- Check the required roles for the accessed Web resource.
- Check the authorization table to make sure that the user, or the groups to which the user belongs, is assigned to one of the required roles.
- View required roles for the Web resource in the deployment descriptor of the Web resource.
- View the authorization table for the application that contains the Web resource, using the administrative console.
- Test with a user who is granted the required roles, to see if the user can access the problem resources.
- If the user is required to have one or more of the required roles, use the administrative console to assign that user to required roles, stop, and restart the application.

If the user is granted required roles, but still fails to access the secured resources, enable security trace, using com.ibm.ws.security.*=all=enabled as the trace specification. Collect trace information for further resolution.

Authentication fails when code pages differ between the client and the server

When a client uses a code page that is different from the server, and non-US-ASCII characters are used for the user ID and password during basic authentication, the login does not succeed. The HTTP header does not include the encoding method information that is necessary to translate the encoded data, so the server does not know how to decode the information correctly.

Use a login form that relies on POST parameters, which are in the HTML body text. The encoding for the text is sent by the browser and so is capable of being decoded properly.

Note: Web services customers are not able to use form login to resolve this problem. Users must ensure there is consistency in the code pages between the client and the server.

Error Message: `CWSCJ0314E: Current Java 2 security policy reported a potential violation on server`

If you find errors on your server similar to:

```
Error Message: CWSCJ0314E: Current Java 2 Security policy reported a potential violation of
Java 2 Security Permission. Please refer to Problem Determination Guide for further information.
{0}Permission/:{1}Code/:{2}{3}Stack Trace/:{4}Code Base Location/:{5}
```

The Java security manager checkPermission method has reported a SecurityException exception .

The reported exception might be critical to the secure system. Turn on security trace to determine the potential code that might have violated the security policy. Once the violating code is determined, verify if the attempted operation is permitted with respect to Java 2 Security, by examining all applicable Java 2 security policy files and the application code.

A more detailed report is enabled by either configuring RAS trace into debug mode, or specifying a Java property.

- Check the trace enabling section for instructions on how to configure Reliability Availability Serviceability (RAS) trace into debug mode, or
- Specify the following property in the **Application Servers >** *server_name* **> ProcessDefinition > Java Virtual Machine** panel from the administrative console in the **Generic JVM arguments** panel:
  - Add the **java.security.debug** run-time flag
  - Valid values:

    access
    Print all debug information including required permission, code, stack, and code base location.

    stack Print debug information including required permission, code, and stack.

    failure Print debug information including required permission, and code.

For a review of Java security policies, see the Java 2 Security documentation at http://java.sun.com/j2se/1.3/docs/guide/security/index.html.

Tip: If the application is running with a Java Mail application programming interface (API), this message might be benign. You can update the *installed Enterprise Application root*/META-INF/was.policy file to grant the following permissions to the application:
- permission java.io.FilePermission ″${user.home}${/}.mailcap″, ″read″;
- permission java.io.FilePermission ″${user.home}${/}.mime.types″, ″read″;
- permission java.io.FilePermission ″${java.home}${/}lib${/}mailcap″, ″read″;
- permission java.io.FilePermission ″${java.home}${/}lib${/}mime.types″, ″read″;

Error message: `CWMSG0508E: The JMS Server security service was unable to authenticate user ID:″ error displayed in SystemOut.log when starting an application server`

This error can result from installing the Java Message Service (JMS) API sample and then enabling security. You can follow the instructions in the Configure and Run page of the corresponding JMS sample documentation to configure the sample to work with WebSphere Application Server security.

You can verify the installation of the message-driven bean sample by launching the installation program, selecting **Custom**, and browsing the components which are already installed in the Select the features you like to install panel. The JMS sample is shown as **Message-Driven Bean Sample**, under Embedded Messaging.

You can also verify this installation by using the administrative console to open the properties of the application server that contains the samples. Select **MDBSamples** and click uninstall.

Error message: `CWSCJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available` after enabling security and starting the application server

This error message can result from selecting Lightweight Third Party Authentication (LTPA) as the authentication mechanism, but not generating the LTPA keys. The LTPA keys encrypt the LTPA token.

To resolve this problem:
1. Click **Security > Global security > Authentication > Authentication mechanisms and expiration> LTPA**
2. Enter a password, which can be anything.
3. Enter the same password in **Confirm Password**.
4. Click **Apply**.
5. Click **Generate Keys**.
6. Click **Save**.

The AccessControlException exception, is reported in the SystemOut.log

The problem is related to the Java 2 security feature of WebSphere Application Server, the API-level security framework that is implemented in WebSphere Application Server. An exception similar to the following example displays. The error message and number can vary.

```
CWSRV0020E: [Servlet Error]-[validator]: Failed to load servlet:
java.security.AccessControlException: access denied
(java.io.FilePermission
app_server_root/systemApps/isclite.ear/isclite.war/WEB-INF/validation.xml read)
```

For an explanation of Java 2 security, how and why to enable or disable it, how it relates to policy files, and how to edit policy files, see the Java 2 security topic in the *Securing applications and their environment* PDF book.The topic explains that Java 2 security is not only used by this product, but developers can also implement it for their business applications. Administrators might need to involve developers, if this exception is created when a client tries to access a resource that is hosted by WebSphere Application Server.

Possible causes of these errors include:

- Syntax errors in a policy file.
- Syntax errors in permission specifications in the ra.xml file that is bundled in a .rar file. This case applies to resource adapters that support connector access to CICS® or other resources.
- An application is missing the specified permission in a policy file, or in permission specifications in ra.xml file bundled in a .rar file.
- The class path is not set correctly, preventing the permissions for the resource.xml file for Service Provider Programming Interface (SPI) from being correctly created.
- A library called by an application, or the application, is missing a doPrivileged block to support access to a resource.
- Permission is specified in the wrong policy file.

To resolve these problems:

- Check all of the related policy files to verify that the permission shown in the exception, for example java.io.FilePermission, is specified.
- Look for a related ParserException exception in the SystemOut.log file which reports the details of the syntax error.

  ```
  CWSCJ0189E: Caught ParserException while creating template for application policy

  profile_root/config/cells/cell_name/nodes/node_name/app.policy
  ```

  Where:

  – *cell_name* represents the name of your cell.
  – *profile_name* represents the name of your profile.
  – *node_name* represents the name of your node.

  The exception is com.ibm.ws.security.util.ParserException: line 18: expected ';', found 'grant'
- Look for a message similar to: `CWSCJ0325W: The permission` *`permission`* `specified in the policy file is unresolved.`
- Check the call stack to determine which method does not have the permission. Identify the class path of this method. If it is hard to identify the method, enable the Java2 security Report.
  - Configuring RAS trace by specifying `com.ibm.ws.security.core.*=all=enabled`, or specifying a Java **property.java.security.debug** property. Valid values for the **java.security.debug** property are:

    access
    Print all debug information including: required permission, code, stack, and code base location.

    stack Print debug information including: required permission, code, and stack.

    failure Print debug information including: required permission and code.
  - The report shows:

    Permission
    The missing permission.

    Code Which method has the problem.

    Stack Trace
    Where the access violation occurred.

    CodeBaseLocation
    The detail of each stack frame.

    Where:

    - *app1* represents the name of your application.
    - *app_server_root* represents the installation root directory for WebSphere Application Server Network Deployment.
    - *profile_root* represents the location and name of a particular profile in your system.
    - *profile1* or *profile_name* represents the name of your profile.
    - *server1* or *server_name*represents the name of your application server.

- If the method is SPI, check the resources.xml file to ensure that the class path is correct.
- To confirm that all of the policy files are loaded correctly, or what permission each class path is granted, enable the trace with **com.ibm.ws.security.policy.*=all=enabled**. All loaded permissions are listed in the trace.log file. Search for the app.policy, was.policy and ra.xml files. To check the permission list for a class path, search for **Effective Policy for *classpath***.
- If there are any syntax errors in the policy file or the ra.xml file, correct them with the policy tool. Avoid editing the policy manually, because syntax errors can result. For additional information about using this tool, refer to the section Using PolicyTool to edit policy files in the *Developing and deploying applications* PDF book.
- If a permission is listed as Unresolved, it does not take effect. Verify that the specified permission name is correct.
- If the class path that is specified in the resource.xml file is not correct, correct it.
- If a required permission does not exist in either the policy files or the ra.xml file, examine the application code to see if you need to add this permission. If so, add it to the proper policy file or the ra.xml file.
- If the permission is not granted outside of the specific method that is accessing this resource, modify the code needs to use a doPrivileged block.
- If this permission does exist in a policy file or a ra.xml file and the permission was loaded correctly, but the class path still does not have the permission in its list, the location of the permission might not be correct. Read the Java 2 security chapter in the *Securing applications and their environment* PDF book carefully to determine in which policy file or ra.xml file to specify that permission.

Tip: If the application is running with the Java Mail API, you can update the *installed Enterprise Application root*/META-INF/was.policy file to grant the following permissions to the application:
- permission java.io.FilePermission ″${user.home}${/}.mailcap″, ″read″;
- permission java.io.FilePermission ″${user.home}${/}.mime.types″, ″read″;
- permission java.io.FilePermission ″${java.home}${/}lib${/}mailcap″, ″read″;
- permission java.io.FilePermission ″${java.home}${/}lib${/}mime.types″, ″read″;

Error Message: `CWSCJ0336E: Authentication failed for user {0} because of the following exception {1}`

This error message results if the user ID that is indicated is not found in the Lightweight Directory Access Protocol (LDAP) user registry. To resolve this problem:
1. Verify that your user ID and password are correct.
2. Verify that the user ID exists in the registry.
3. Verify that the base distinguished name (DN) is correct.
4. Verify that the user filter is correct.
5. Verify that the bind DN and the password for the bind DN are correct. If the bind DN and password are not specified, add the missing information and retry.
6. Verify that the host name and LDAP type are correct.

Consult with the administrator of the user registry if the problem persists.

Error Message: `SECJ0352E: Could not get the users matching the pattern {0} because of the following exception {1}`

This authentication failure message displays when an external user account repository is corrupted or unavailable, and WebSphere Application Server is unable to authenticate the user name in the repository. Generally, authentication error messages are followed by additional information that indicates the nature or root cause of the problem, such as:

```
Make sure the users matching the pattern exist in the registry. Contact your service representative if the problem persists.
```

This additional information might not provide a clear user action if the user account repository is corrupted or the user loses connectivity between WebSphere Application Server and an external user account repository. The external user account repository, which is referred to as a repository in this document, might be a Lightweight Directory Access Protocol (LDAP) product.

To resolve this problem, you might need to re-install the repository and verify that it installs successfully by testing the connection.

Note: Proceed with the following steps only if you have ensured that all WebSphere Application Server-related configuration settings are accurate.

Complete the following steps to resolve the issue:

1. Restart both the repository and WebSphere Application Server.
2. Test the connection to the repository. If the connection attempt still fails, it might be necessary to re-install the repository.
3. If diagnostics are provided with the repository, run them to avoid having to re-install the repository.

   Note: If the previous steps do not fix the problem, you might need to re-install the repository. Before proceeding, generate a complete list of all the configured users and groups; you will need to re-populate these fields after the re-installation.
4. If necessary, re-install the corrupted repository.
5. Populate the users and groups from your list into the newly installed repository.
6. Restart both the repository and WebSphere Application server.
7. In the administrative console, navigate to **Security > Global security**, and select the appropriate user account repository. For example, select **Standalone LDAP registry** if you are using a standalone Lightweight Directory Access Protocol repository.
8. Click **Test connection** to ensure that WebSphere Application Server can connect to the repository.

Generate keys error when using the Profile Management tool to create a new profile

When you create a new profile using either the Profile Management tool or the command-line manageprofiles utility, an error message displays that indicates either partial success or failure. The error message, which is located in the *install_dir*/logs/manageprofiles/*profile_name*_create.log file, might point to an error in either the generateKeysforSingleProfile task or the generateKeysForCellProfile task.

The Profile Creation tool and the manageprofiles utility invoke several tasks. The generateKeysForSingleProfile task is invoked when you create a stand-alone application server or a deployment manager profile. The generateKeysForCellProfile task is invoked when you create a cell profile. Both of these tasks are the first tasks to invoke the wsadmin commands. Although the log indicates an error in one of these tasks, the error might actually result from a wsadmin command failure and not an error in the security tasks.

To determine the actual cause of the problem, review the information that is provided in the following log files:

- *install_dir*/logs/manageprofiles/*profile_name*_create.log file indicates the error code of the failure
- *install_dir*/logs/manageprofiles/*profile_name*/keyGeneration.log file
- *install_dir*/logs/manageprofiles/*profile_name*/wsadminListener.log file

Access problems after enabling security

Use this information if you are experiencing access problems after enabling security.

What kind of error are you seeing?

- "I cannot access all or part of the administrative console or use the wsadmin tool after enabling security" on page 107
- "I cannot access a Web page after enabling security" on page 107
- "Authentication error accessing a Web page" on page 101
- "Authorization error accessing a Web page" on page 101
- "The client cannot access an enterprise bean after enabling security" on page 108

- "Client program never gets prompted when accessing secured enterprise bean" on page 110
- "Cannot stop an application server, node manager, or node after enabling security" on page 110
- "The AccessControlException exception, is reported in the SystemOut.log" on page 103
- "After enabling single sign-on, I cannot logon to the administrative console" on page 110
- "The following exception displays in the SystemOut.log file after I start the server and enable security: ″SECJ0306E: No received or invocation credential exists on the thread.″ " on page 110
- "A Name NotFoundException error occurs when initially connecting to the federated repositories." on page 111

For general tips on diagnosing and resolving security-related problems, see the topic "Security components troubleshooting tips" on page 87.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, see Troubleshooting help from IBM.

I cannot access all or part of the administrative console or use the wsadmin tool after enabling security

- If you cannot access the administrative console, or view and update certain objects, look in the SystemOut log of the application server which hosts the administrative console page for a related error message.
- You might not have authorized your ID for administrative tasks. This problem is indicated by errors such as:
  - `[8/2/02 10:36:49:722 CDT] 4365c0d9 RoleBasedAuth A CWSCJ0305A: Role based authorization check failed for security name MyServer/myUserId, accessId MyServer/S-1-5-21-882015564-4266526380-2569651501-1005 while invoking method getProcessType on resource Server and module Server.`
  - Exception message: ″`CWWMN0022E: Access denied for the getProcessType operation on Server MBean`″
  - `When running the command: wsadmin -username j2ee -password j2ee: CWWAX7246E: Cannot establish ″SOAP″ connection to host ″BIRKT20″ because of an authentication failure. Ensure that user and password are correct on the command line or in a properties file.`

  To grant an ID administrative authority, from the administrative console, click **System Administration > Console Users** and validate that the ID is a member. If the ID is not a member, add the ID with at least monitor access privileges, for read-only access.
- Verify that the enable_trusted_application flag is set to `true`. To check the enable_trusted_application flag value using the administrative console, click **Security > Global security**. Under Additional properties, click **Custom properties > EnableTrustedApplications**.

I cannot access a Web page after enabling security

When secured resources are not accessible, probable causes include:

- Authentication errors - WebSphere Application Server security cannot identify the ID of the person or process. Symptoms of authentication errors include:

  On a Netscape browser:
  - `Authorization failed. Retry?` message is displayed after an attempt to log in.
  - Accepts any number of attempts to retry login and displays `Error 401` message when Cancel is clicked to stop retry.
  - A typical browser message displays: `Error 401: Basic realm='Default Realm'`.

  On an Internet Explorer browser:
  - Login prompt displays again after an attempt to log in.
  - Allows three attempts to retry login.
  - Displays `Error 401` message after three unsuccessful retries.
- Authorization errors - The security function has identified the requesting person or process as not authorized to access the secured resource. Symptoms of authorization errors include:
  - Netscape browser: ″Error 403: AuthorizationFailed″ message is displayed.

- Internet Explorer:
  - ″You are not authorized to view this page″ message is displayed.
  - ″HTTP 403 Forbidden″ error is also displayed.
- SSL errors - WebSphere Application Server security uses Secure Sockets Layer (SSL) technology internally to secure and encrypt its own communication, and incorrect configuration of the internal SSL settings can cause problems. Also you might have enabled SSL encryption for your own Web application or enterprise bean client traffic which, if configured incorrectly, can cause problems regardless of whether WebSphere Application Server security is enabled.
  - SSL-related problems are often indicated by error messages that contain a statement such as: `ERROR: Could not get the initial context or unable to look up the starting context.Exiting.` followed by `javax.net.ssl.SSLHandshakeException`

The client cannot access an enterprise bean after enabling security

If the client access to an enterprise bean fails after security is enabled:

- Review the steps for securing and granting access to resources.
- Browse the server JVM logs for errors relating to enterprise bean access and security. Look up any errors in the message table.

  Errors similar to `Authorization failed for /UNAUTHENTICATED while invoking` *`resource`* `securityName:/UNAUTHENTICATED;accessId:UNAUTHENTICATED not granted any of the required roles` *`roles`* indicate that:
  - An unprotected servlet or JavaServer Pages (JSP) file accessed a protected enterprise bean. When an unprotected servlet is accessed, the user is not prompted to log in and the servlet runs as UNAUTHENTICATED. When the servlet makes a call to an enterprise bean that is protected, the servlet fails.

    To resolve this problem, secure the servlet that is accessing the protected enterprise bean. Make sure that the runAs property for the servlet is set to an ID that can access the enterprise bean.
  - An unauthenticated Java client program is accessing an enterprise bean resource that is protected. This situation can happen if the file that is read by the `sas.client.props` properties file that is used by the client program does not have the `securityEnabled` flag set to `true`.

    To resolve this problem, make sure that the `sas.client.props` file on the client side has its securityEnabled flag set to `true`.

  Errors similar to Authorization failed for *`valid_user`* `while invoking` *`resource`* `securityName:/username;accessId:xxxxxx not granted any of the required roles` *`roles`* indicate that a client attempted to access a secured enterprise bean resource, and the supplied user ID is not assigned the required roles for that enterprise bean.
  - Check that the required roles for the enterprise bean resource are accessed. View the required roles for the enterprise bean resource in the deployment descriptor of the Web resource.
  - Check the authorization table and make sure that the user or the group that the user belongs to is assigned one of the required roles. You can view the authorization table for the application that contains the enterprise bean resource using the administrative console.

If `org.omg.CORBA.NO_PERMISSION` exceptions occur when programmatically logging on to access a secured enterprise bean, an authentication exception has occurred on the server. Typically the CORBA exception is triggered by an underlying `com.ibm.WebSphereSecurity.AuthenticationFailedException`. To determine the actual cause of the authentication exception, examine the full trace stack:

1. Begin by viewing the text following `WSSecurityContext.acceptSecContext(), reason:` in the exception. Typically, this text describes the failure without further analysis.
2. If this action does not describe the problem, look up the Common Object Request Broker Architecture (CORBA) minor code. The codes are listed in the article titled Troubleshooting the security components reference.

   For example, the following exception indicates a CORBA minor code of 49424300. The explanation of this error in the CORBA minor code table reads:

   ```
   authentication failed error
   ```

In this case the user ID or password supplied by the client program is probably not valid:

```
org.omg.CORBA.NO_PERMISSION: Caught WSSecurityContextException in
WSSecurityContext.acceptSecContext(), reason: Major Code[0] Minor Code[0]
Message[ Exception caught invoking authenticateBasicAuthData from SecurityServer
for user jdoe. Reason: com.ibm.WebSphereSecurity.AuthenticationFailedException]
minor code: 49424300 completed:
No at com.ibm.ISecurityLocalObjectBaseL13Impl.PrincipalAuthFailReason.map_auth_fail_to_minor_code
(PrincipalAuthFailReason.java:83)
```

A `CORBA INITIALIZE` exception with `CWWSA1477W: SECURITY CLIENT/SERVER CONFIGURATION MISMATCH` error embedded, is received by client program from the server.

This error indicates that the security configuration for the server differs from the client in some fundamental way. The full exception message lists the specific mismatches. For example, the following exception lists three errors:

```
Exception received: org.omg.CORBA.INITIALIZE:
CWWSA1477W: SECURITY CLIENT/SERVER CONFIG MISMATCH:
The client security configuration (sas.client.props or outbound settings in
administrative console) does not support the server security configuration for
the following reasons:
ERROR 1: CWWSA0607E: The client requires SSL Confidentiality but the server does not
         support it.
ERROR 2: CWWSA0610E: The server requires SSL Integrity but the client does not
         support it.
ERROR 3: CWWSA0612E: The client requires client (e.g., userid/password or token),
         but the server does not support it.
     minor code: 0
     completed: No at
com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityConnectionInterceptor.getConnectionKey
(SecurityConnectionInterceptor.java:1770)
```

In general, resolving the problem requires a change to the security configuration of either the client or the server. To determine which configuration setting is involved, look at the text following the `CWWSA` error message. For more detailed explanations and instructions, look in the message reference, by selecting the **Reference** view of the information center navigation and expanding **Messages** in the navigation tree.

In these particular cases:

- In ERROR 1, the client is requiring SSL confidentiality but the server does not support SSL confidentiality. Resolve this mismatch in one of two ways. Either update the server to support SSL confidentiality or update the client so that it no longer requires it.
- In ERROR 2, the server requires SSL integrity but the client does not support SSL integrity. Resolve this mismatch in one of two ways. Either update the server to support SSL integrity or update the client so that it no longer requires it.
- In ERROR 3, the client requires client authentication through a user id and password, but the server does not support this type of client authentication. Either the client or the server needs to change the configuration. To change the client configuration, modify the `SAS.CLIENT.PROPS` file for a pure client or change the outbound configuration for the server in the Security administrative console. To change the configuration for the target server, modify the inbound configuration in the Security administrative console.

Similarly, an exception like `org.omg.CORBA.INITIALIZE: JSAS0477W: SECURITY CLIENT/SERVER CONFIG MISMATCH:` appearing on the server trying to service a client request indicates a security configuration mismatch between client and server. The steps for resolving the problem are the same as for the JSAS1477W exceptions previously described.

Client program never gets prompted when accessing secured enterprise bean

Even though it seems that security is enabled and an enterprise bean is secured, occasions can occur when the client runs the remote method without prompting. If the remote method is protected, an authorization failure results. Otherwise, run the method as an unauthenticated user.

Possible reasons for this problem include:
- The server with which you are communicating might not have security enabled. Check with the WebSphere Application Server administrator to ensure that the server security is enabled. Access the security settings from within the **Security** section of the administrative console.
- The client does not have security enabled in the `sas.client.props` file. Edit the `sas.client.props` file to ensure the property com.ibm.CORBA.securityEnabled is set to `true`.
- The client does not have a ConfigURL specified. Verify that the property com.ibm.CORBA.ConfigURL is specified on the command line of the Java client, using the -D parameter.
- The specified ConfigURL does not have a valid URL syntax, or the `sas.client.props` that is pointed to cannot be found. Verify that the com.ibm.CORBA.ConfigURL property is valid. Check the Java documentation for a description of URL formatting rules. Also, validate that the file exists at the specified path.
- The client configuration does not support message layer client authentication (user ID and password). Verify that the `sas.client.props` file has one of the following properties set to `true`:
  - `com.ibm.CSI.performClientAuthenticationSupported=true`
  - `com.ibm.CSI.performClientAuthenticationRequired=true`
- The server configuration does not support message layer client authentication, which consists of a user ID and password. Check with the WebSphere Application Server administrator to verify that user ID and password authentication is specified for the inbound configuration of the server within the System Administration section of the administrative console administration tool.

Cannot stop an application server, node manager, or node after enabling security

If you use command-line utilities to stop WebSphere Application Server processes, apply additional parameters after enabling security to provide authentication and authorization information.

Use the `./stopServer -help` command to display the parameters to use.

Use the following command options after enabling security:
- `./stopServer serverName -username` *name* `-password` *password*
- `./stopNode -username` *name* `-password` *password*
- `./stopManager -username` *name* `-password` *password*

After enabling single sign-on, I cannot logon to the administrative console

This problem occurs when single sign-on (SSO) is enabled, and you attempt to access the administrative console using the short name of the server, for example `http://myserver:`*port_number*`/ibm/console`. The server accepts your user ID and password, but returns you to the logon page instead of the administrative console.

To correct this problem, use the fully qualified host name of the server, for example `http://myserver.mynetwork.mycompany.com:9060/ibm/console`.

The following exception displays in the SystemOut.log file after I start the server and enable security: ″SECJ0306E: No received or invocation credential exists on the thread.″

The following message displays when one or more nodes within the cell was not synchronized during configuration:

```
SECJ0306E: No received or invocation credential exists on the thread. The Role based
authorization check will not have an accessId of the caller to check. The parameters
are: access check method getServerConfig on resource FileTransferServer and module
FileTransferServer. The stack trace is java.lang.Exception: Invocation and received
credentials are both null.
```

Make sure that each of the nodes are synchronized and then restart the deployment manager.

A Name NotFoundException error occurs when initially connecting to the federated repositories.

When the server attempts an indirect lookup on the java:comp/env/ds/wimDS name and makes its initial EJB connection to the federated repositories, the following error message displays in the SystemOut.log file:

```
NMSV0612W: A NameNotFound Exception
```

The NameNotFoundException error is caused by the reference binding definition for the jdbc/wimDS Java Naming and Directory interface (JNDI) name in the ibm-ejb-jar-bnd.xmi file. You can ignore this warning message. The message does not display when the wimDS database repository is configured.

Secure Sockets Layer errors

You might encounter various problems after configuring or enabling Secure Sockets Layer (SSL). You may not be able to stop the deployment manager after configuring the SSL. You may not be able to access resource using HTTPS. The client and the server may not be able to negotiate the proper level of security. The problems mentioned here are only a few of the possibilities. Solving these problems is imperative to the successful operation of WebSphere Application Server.

What type of problem are you having?

- “Stopping the deployment manager after configuring Secure Sockets Layer”
-
- “javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure” on page 112
- “javax.net.ssl.SSLHandshakeException: unknown certificate” on page 112
- “javax.net.ssl.SSLHandshakeException: bad certificate” on page 113
- “org.omg.CORBA.INTERNAL: EntryNotFoundException or NTRegistryImp E CWSCJ0070E: No privilege id configured for: error when programmatically creating a credential” on page 113
- “″Catalog″ tablet is blank (no item displayed) in GUI application client” on page 114
- “Modifying SSL Configurations after migration using -scriptCompatibility true” on page 114

Stopping the deployment manager after configuring Secure Sockets Layer

After configuring the Secure Sockets Layer repertoires, if you stop the deployment manager without also stopping the node agents, you might receive the following error message when you restart the deployment manager:

```
CWWMU0509I: The server "nodeagent" cannot be reached. It appears to be stopped.
CWWMU0211I: Error details may be seen in the file:
          /opt/WebSphere/AppServer/logs/nodeagent/stopServer.log
```

The error occurs because the deployment manager did not propagate the new SSL certificate to the node agents. The node agents are using an older certificate file than the deployment manager and the certificate files are incompatible. To work around this problem, you must manually stop the node agent and deployment manager processes.

You need to consider certain items when identifying the specific process to stop. For each process that is stopped, WebSphere Application Server stores the process ID in a `pid` file and you need to find these

`*.pid` files. For example, the `server1.pid` for a stand-alone install action might be found at: *app_server_root*`/logs/server1.pid`

javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure

If you see a Java exception stack similar to the following example:

```
[Root exception is org.omg.CORBA.TRANSIENT:  CAUGHT_EXCEPTION_WHILE_CONFIGURING_
SSL_CLIENT_SOCKET: CWWJE0080E:   javax.net.ssl.SSLHandshakeException - The client
and server could not negotiate the desired level of   security. Reason: handshake
failure:host=MYSERVER,port=1079 minor code: 4942F303 completed: No]   at
com.ibm.CORBA.transport.TransportConnectionBase.connect
(TransportConnectionBase.java:NNN)
```

Some possible causes are:
- Not having common ciphers between the client and server.
- Not specifying the correct protocol.

To correct these problems:
1. Review the SSL settings. In the administrative console, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations >** *endpoint_configuration_name*. Under Related items, click **SSL configurations >** *SSL_configuration_name*. You can also browse the file manually by viewing the *app_server_root*/`properties/sas.client.props` file.
2. Check the property that is specified by the `com.ibm.ssl.protocol` file to determine which protocol is specified.
3. Check the cipher types that are specified by the com.ibm.ssl.enabledCipherSuites interface. You might want to add more cipher types to the list. To see which cipher suites are currently enabled, click **Quality of protection settings (QoP)**, and look for the **Cipher Suites** property.
4. Correct the protocol or cipher problem by using a different client or server protocol and cipher selection. Typical protocols are SSL or SSLv3.
5. Make the cipher selection 40-bit instead of 128-bit. For Common Secure Interoperability Version 2 (CSIv2), set both of the following properties to false in the sas.client.props file, or set security level=medium in the administrative console settings:
   - com.ibm.CSI.performMessageConfidentialityRequired=false
   - com.ibm.CSI.performMessageConfidentialitySupported=false

javax.net.ssl.SSLHandshakeException: unknown certificate

If you see a Java exception stack similar to the following example, it might be caused by not having the personal certificate for the server in the client truststore file:

```
ERROR: Could not get the initial context or unable to look up the starting context.
Exiting.  Exception received: javax.naming.ServiceUnavailableException: A
communication failure occurred while attempting to obtain an initial context using
the provider url: "corbaloc:iiop:localhost:2809". Make sure that the host and port
information is correct and that the server identified by the provider url is a
running name server. If no port number is specified, the default port number 2809
is used. Other possible causes include the network environment or workstation
network configuration. [Root exception is org.omg.CORBA.TRANSIENT:
CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: CWWJE0080E:
javax.net.ssl.SSLHandshakeException - The client and server could not
negotiate the desired level of security. Reason: unknown
certificate:host=MYSERVER,port=1940 minor code: 4942F303 completed: No]
```

To correct this problem:
1. Check the client truststore file to determine if the signer certificate from the server personal certificate is there. For a self-signed server personal certificate, the signer certificate is the public key of the

personal certificate. For a certificate authority (CA)-signed server personal certificate, the signer certificate is the root CA certificate of the CA that signed the personal certificate.

2. Add the server signer certificate to the client truststore file.

javax.net.ssl.SSLHandshakeException: bad certificate

A Java exception stack error might display if the following situations occur:

- A personal certificate exists in the client keystore that is used for SSL mutual authentication.
- The signer certificate is not extracted into the server truststore file, and thus the server cannot trust the certificate whenever the SSL handshake is made.

The following message is an example of the Java exception stack error:

```
ERROR: Could not get the initial context or unable to look
up the starting context. Exiting.
Exception received: javax.naming.ServiceUnavailableException:
A communication failure occurred while attempting to obtain an
initial context using the provider url: "corbaloc:iiop:localhost:2809".
Make sure that the host and port information is correct and that the
server identified by the provider url is a running name
server. If no port number is specified, the default port number 2809
is used. Other possible causes include the network environment or
workstation network configuration.
[Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_
CLIENT_SOCKET: CWWJE0080E: javax.net.ssl.SSLHandshakeException - The client and
server could not negotiate the desired level of security. Reason:
bad certificate: host=MYSERVER,port=1940 minor code: 4942F303 completed: No]
```

To verify this problem, check the server truststore file to determine if the signer certificate from the client personal certificate is there. For a self-signed client personal certificate, the signer certificate is the public key of the personal certificate. For a certificate authority-signed client personal certificate, the signer certificate is the root CA certificate of the CA that signed the personal certificate.

To correct this problem, add the client signer certificate to the server truststore file.

org.omg.CORBA.INTERNAL: EntryNotFoundException or NTRegistryImp E CWSCJ0070E: No privilege id configured for: error when programmatically creating a credential

If you encounter the following exception in a client application attempting to request a credential from a WebSphere Application Server using SSL mutual authentication:

```
ERROR: Could not get the initial context or unable to look up the starting context.
Exiting. Exception received: org.omg.CORBA.INTERNAL: Trace from server: 1198777258
at host MYHOST on port 0 >>org.omg.CORBA.INTERNAL: EntryNotFoundException minor
code: 494210B0 completed:
No at com.ibm.ISecurityLocalObjectBaseL13Impl.PrincipalAuthFailReason.
map_auth_fail_to_minor_code(PrincipalAuthFailReason.java:99)
```

or a simultaneous error from the WebSphere Application Server that resembles:

```
[7/31/02 15:38:48:452 CDT] 27318f5 NTRegistryImp E CWSCJ0070E: No privilege id
configured for: testuser
```

The cause might be that the user ID sent by the client to the server is not in the user registry for that server.

To confirm this problem, check that an entry exists for the personal certificate that is sent to the server. Depending on the user registry mechanism, look at the native operating system user ID or Lightweight Directory Access Protocol (LDAP) server entries.

To correct this problem, add the user ID to the user registry entry (for example, operating system, LDAP directory, or other custom registry) for the personal certificate identity.

″Catalog″ tablet is blank (no item displayed) in GUI application client

This error message occurs when you install an ActiveX client sample application that uses the PlantsByWebSphere Active X to EJB Bridge.

The cause is that the server certificate is not in the client trustore that is specified in the client.ssl.props file. Although the ″com.ibm.ssl.enableSignerExchangePrompt″ signer property might be set to `true`, the auto-exchange prompt only supports a command-line prompt. If the sample application relies on a graphical user interface and does not provide access to a command prompt, for example using standard in and standard out, the auto-exchange prompt does not function.

Note: The applet client under the Client Technology Samples does not have access to the command prompt and it cannot see the auto-exchange prompt. Thus, the applet client cannot rely on the auto-exchange prompt feature.

To correct this problem, retrieve the certificate manually using the retrieveSigners utility.

Modifying SSL Configurations after migration using -scriptCompatibility true

After migrating using scriptCompatibility true, all attributes of the SSL configurations cannot be edited through the administrative console. In particular, the hardware cryptography settings cannot be displayed or edited.

By using the scriptCompatibility true flag, the SSL configurations are not migrated to the new format for support in the 6.1 release. New capabilities were added that are not supported when the configurations are not migrated to the latest format.

To solve this problem, create new SSL configuration definitions to replace those in the pre 6.1 format or continue to edit those configurations using scripting.

Single sign-on configuration troubleshooting tips

Several common problems can occur when you configure single sign-on (SSO) between a WebSphere Application Server and a Domino® server. Some such problems are: Failure to save the Domino Web SSO configuration, authentication failures when accessing a protected resource, and SSO failures when accessing a protected resource. You can take some actions to correct these error situations and restore the SSO.

- Failure to save the Domino Web SSO configuration document

  The client must find Domino server documents for the participating SSO Domino servers. The Web SSO configuration document is encrypted for the servers that you specify. The home server that is indicated by the client location record must point to a server in the Domino domain where the participating servers reside. This pointer ensures that lookups can find the public keys of the servers.

  If you receive a message stating that one or more of the participating Domino servers cannot be found, then those servers cannot decrypt the Web SSO configuration document or perform SSO.

  When the Web SSO configuration document is saved, the status bar indicates how many public keys are used to encrypt the document by finding the listed servers, authors, and administrators in the document.
- Failure of the Domino server console to load the Web SSO configuration document at Domino HTTP server startup

  During configuration of SSO, the server document is configured for Multi-Server in the **Session Authentication** field. The Domino HTTP server tries to find and load a Web SSO configuration document during startup. The Domino server console reports the following information if a valid document is found and decrypted: `HTTP: Successfully loaded Web SSO Configuration`.

If a server cannot load the Web SSO configuration document, SSO does not work. In this case, a server reports the following message: `HTTP: Error Loading Web SSO configuration`. Reverting to single-server session authentication.

Verify that only one Web SSO configuration document is in the Web configurations view of the Domino directory and in the $WebSSOConfigs hidden view. You cannot create more than one document, but you can insert additional documents during replication.

If you can verify only one Web SSO configuration document, consider another condition. When the public key of the server document does not match the public key in the ID file, this same error message can display. In this case, attempts to decrypt the Web SSO configuration document fail and the error message is generated.

This situation can occur when the ID file is created multiple times, but the Server document is not updated correctly. Usually, an error message is displayed on the Domino server console stating that the public key does not match the server ID. If this situation occurs, SSO does not work because the document is encrypted with a public key for which the server does not possess the corresponding private key.

To correct a key-mismatch problem:

1. Copy the public key from the server ID file and paste it into the Server document.
2. Create the Web SSO configuration document again.

- Authentication fails when accessing a protected resource.

  If a Web user is repeatedly prompted for a user ID and password, SSO is not working because either the Domino or the WebSphere Application Server security server cannot authenticate the user with the Lightweight Directory Access Protocol (LDAP) server. Check the following possibilities:

  – Verify that the LDAP server is accessible from the Domino server machine. Use the TCP/IP ping utility to check TCP/IP connectivity and to verify that the host machine is running.

  – Verify that the LDAP user is defined in the LDAP directory. Use the **idsldapsearch** utility to confirm that the user ID exists and that the password is correct. For example, you can run the following command, entered as a single line:

    You can use the OS/400® Qshell, a UNIX® shell, or a Windows DOS prompt

```
% ldapsearch -D "cn=John Doe, ou=Rochester, o=IBM, c=US" -w mypassword
-h myhost.mycompany.com -p 389 -b "ou=Rochester, o=IBM, c=US" (objectclass=*)
```

  The percent character (**%**) indicates the prompt and is not part of the command. A list of directory entries is expected. Possible error conditions and causes are contained in the following list:

  - No such object: This error indicates that the directory entry referenced by either the user's distinguished name (DN) value, which is specified after the **-D** option, or the base DN value, which is specified after the **-b** option, does not exist.
  - Credentials that are not valid: This error indicates that the password is not valid.
  - Cannot contact the LDAP server: This error indicates that the host name or the port specified for the server is not valid or that the LDAP server is not running.
  - An empty list means that the base directory that is specified by the **-b** option does not contain any directory entries.

  – If you are using the user's short name or user ID instead of the distinguished name, verify that the directory entry is configured with the short name. For a Domino directory, verify the **Short name/UserID** field of the Person document. For other LDAP directories, verify the userid property of the directory entry.

  – If Domino authentication fails when using an LDAP directory other than a Domino directory, verify the configuration settings of the LDAP server in the Directory assistance document in the Directory assistance database. Also verify that the Server document refers to the correct Directory assistance document. The following LDAP values that are specified in the Directory Assistance document must match the values specified for the user registry in the WebSphere Application Server administrative domain:

  - Domain name

    - LDAP host name
    - LDAP port
    - Base DN

    Additionally, the rules that are defined in the Directory assistance document must refer to the base distinguished name (DN) of the directory that contains the directory entries of the users.

    You can trace Domino server requests to the LDAP server by adding the following line to the server `notes.ini` file:

    ```
    webauth_verbose_trace=1
    ```

    After restarting the Domino server, trace messages are displayed in the Domino server console as Web users attempt to authenticate to the Domino server.

- Authorization failure when accessing a protected resource.

  After authenticating successfully, if an authorization error message is displayed, security is not configured correctly. Check the following possibilities:
  - For Domino databases, verify that the user is defined in the access-control settings for the database. Refer to the Domino administrative documentation for the correct way to specify the user's DN. For example, for the `DN cn=John Doe, ou=Rochester, o=IBM, c=US`, the value on the access-control list must be set as `John Doe/Rochester/IBM/US`.
  - For resources that are protected by WebSphere Application Server, verify that the security permissions are set correctly.
    - If granting permissions to selected groups, make sure that the user attempting to access the resource is a member of the group. For example, you can verify the members of the groups by using the following Web site to display the directory contents: `Ldap://myhost.mycompany.com:389/ou=Rochester, o=IBM, c=US??sub`
    - If you changed the LDAP configuration information (host, port, and base DN) in a WebSphere Application Server administrative domain since the permissions were set, the existing permissions are probably not valid and need to be recreated.

- SSO failure when accessing protected resources.

  If a Web user is prompted to authenticate with each resource, SSO is not configured correctly. Check the following possibilities:
  1. Configure both WebSphere Application Server and the Domino server to use the same LDAP directory. The HTTP cookie that is used for SSO stores the full DN of the user, for example, `cn=John Doe, ou=Rochester, o=IBM, c=US`, and the domain name service (DNS) domain.
  2. Define Web users by hierarchical names if the Domino directory is used. For example, update the **User name** field in the Person document to include names of this format as the first value: `John Doe/Rochester/IBM/US`.
  3. Specify the full DNS server name, not just the host name or TCP/IP address for Web sites issued to Domino servers and WebSphere Application Servers that are configured for SSO. For browsers to send cookies to a group of servers, the DNS domain must be included in the cookie, and the DNS domain in the cookie must match the Web address. This requirement is why you cannot use cookies across TCP/IP domains.
  4. Configure both Domino and the WebSphere Application Server to use the same DNS domain. Verify that the DNS domain value is exactly the same, including capitalization. You need the name of the DNS domain in which WebSphere Application Server is configured. For additional information about configuring DNS domains for SSO, refer to the Single sign-on topic in the *Securing applications and their environment* PDF book.
  5. Verify that the clustered Domino servers have the host name populated with the full DNS server name in the server document. By using the full DNS server name, Domino Internet Cluster Manager (ICM) can redirect to cluster members using SSO. If this field is not populated, by default, ICM redirects Web addresses to clustered Web servers by using the host name of the server only. ICM cannot send the SSO cookie because the DNS domain is not included in the Web address. To correct the problem:
     1. Edit the Server document.
     2. Click **Internet Protocols > HTTP** tab.

c. Enter the full DNS name of the server in the **Host names** field.

6. If a port value for an LDAP server is specified for a WebSphere Application Server administrative domain, edit the Domino Web SSO configuration document and insert a backslash character (\) into the value of the **LDAP Realm** field before the colon character (**:**). For example, replace `myhost.mycompany.com:389` with `myhost.mycompany.com\:389`.

- Users are not logged out after the HTTP session timer expires.

  If users of WebSphere Application Server log onto an application and sit idle longer than the specified HTTP session timeout value, the user information is not invalidated and user credentials stay active until LTPA token timeout occurs.

  After you apply PK25740, complete the following steps to log out users from the application after the HTTP session has expired.

  1. In the administrative console, click **Security > Global security**.
  2. Under Custom properties, click **New**.
  3. In the Name field, enter com.ibm.ws.security.web.logoutOnHTTPSessionExpire.
  4. In the Values field, enter true.
  5. Click Apply and Save to save the changes to your configuration.
  6. Resynchronize and restart the server.

Enterprise Identity Mapping troubleshooting tips

The following information provides troubleshooting information for Enterprise Identity Mapping (EIM) configuration or connection factory configuration.

AdminControl service is not available

Symptom
The following message is displayed:

Message: WASX7017E: Exception received while running file ″/QIBM/ProdData/OS400/Java400/cfgIdToken.jacl″; exception information: com.ibm.ws.scripting.ScriptingException: AdminControl service not available.

Explanation
The application server or deployment manager of the WebSphere Application Server profile is not started, or the wsadmin option `-conntype NONE` is specified.

Configuration-related messages returned by the sample application to your Web browser session

Symptom
The following message is displayed:

Message: com.ibm.as400.access.AS400SecurityException: User ID is not known.

Explanation
The EIM does not contain a mapping for the user ID that is used to log in to the sample application.

Symptom
The following message is displayed:

Message: com.ibm.as400.access.ServerStartupException: Password encryption indicator is not valid.

Explanation
The target iSeries server is not configured for Enterprise Identity Mapping (EIM).

Symptom
The following message is displayed:

Message: java.net.ConnectException: A remote host refused an attempted connect operation.

Explanation
The target server is not an iSeries server.

Symptom
The following message is displayed:

Message: The lookup for the connection factory failed. Either the connector is not configured, or the servlet resource reference (JNDI name) is not set correctly in the web.xml file. The servlet expects the resource reference in the web.xml file to be eis/ IdentityToken_Shared_Reference.

Explanation
Either the connector is not configured, or the servlet resource reference (JNDI name) is not set correctly in the `web.xml` file. The servlet expects the resource reference in the `web.xml` file to be `eis/IdentityToken_Shared_Reference`.

Symptom
The following message is displayed:

Message: The JAAS Subject object was not passed to the Java 2 Connector (J2C) connector because WebSphere Application Server security is not correctly configured for the servlet.

Explanation
WebSphere Application Server administrative security is not enabled.

Symptom
The following message is displayed:

Message: javax.resource.ResourceException: com.ibm.eim.jndi.DomainJNDI:method_name: failed to connect to initial directory context.

Explanation
This message is caused by one of the following issues:

- The authentication data entry that is configured for the connection factory contains an incorrect Lightweight Directory Access Protocol (LDAP) distinguished name.
- The authentication data entry that is configured for the connection factory contains an incorrect LDAP password.
- The LDAP host name that is configured for the connection factory is incorrect.
- The LDAP port that is configured for the connection factory is incorrect.
- The LDAP server is not started.
- The Enterprise Identity Mapping (EIM) domain name that is configured for the connection factory is incorrect.
- The EIM parent name that is configured for the connection factory is incorrect.

Symptom
The following message is displayed:

Message: javax.resource.ResourceException: Input URL is null or not valid.

Explanation
An LDAP host name is not configured for the connection factory.

Symptom
The following message is displayed:

Message: com.ibm.as400.access.AS400SecurityException: An unknown problem occurred.

Explanation
The target iSeries server is not joined to the EIM domain that is configured for the connection factory, or the EIM source registry name is incorrect.

Perform the following steps to enable trace for EIM:

Note: This trace is only available for `idTokenRA.JCA15.rar`.

1. From the administrative console, select **Servers > Application Servers > server_name > Change Log Details Levels**.
2. Click the **Runtime** tab.
3. Select **Save runtime changes to configuration as well**.
4. Remove any previous entries in the text field, and type the following:

   ```
   com.ibm.jca.idtoken.*=all: com.ibm.eim.token.*=all
   ```
5. Click **Apply** and save the changes.

Security authorization provider troubleshooting tips

This article describes the issues you might encounter using a Java Authorization Contract for Containers (JACC) authorization provider. Tivoli Access Manager is bundled with WebSphere Application Server as an authorization provider. However, you also can plug in your own authorization provider.

Tivoli Access Manager as a Java Authorization Contract for Containers authorization provider

You might encounter the following issues when using Tivoli Access Manager as a JACC authorization provider:

- The configuration of JACC might fail.
- The server might fail to start after configuring JACC.
- The application might not deploy properly.
- The startServer command might fail after you have configured Tivoli Access Manager or a clean uninstall did not take place after unconfiguring JACC.
- An ″HPDIA0202w An unknown user name was presented to Access Manager″ error might occur.
- An ″HPDAC0778E The specified user's account is set to invalid″ error might occur.
- An WASX7017E: Exception received while running file ″InsuranceServicesSingle.jacl″ error might occur.
- "Access denied exceptions accessing applications when using JACC" on page 122
- "An ″HPDBA0219E: An error occurred reading data from an SSL connection″ might occur" on page 122

External providers for Java Authorization Contract for Containers authorization provider

You might encounter the following issues when you use an external provider for JACC authorization:

- An ″HPDJA0506E Invalid argument: Null or zero-length user name field for the ACL entry″ error might occur.

The configuration of JACC might fail

If you have problems configuring JACC, check the following items:

- Ensure that the parameters are correct. For example, you do not want a number after TAM_Policy_server_hostname:7135, but you do want be a number after TAM_Authorization_server_hostname:7136 (for example, TAM_Authorization_server_hostname:7136:1).
- If a message such as "server can't be contacted" is displayed, it is possible that the host names or port numbers of the Tivoli Access Manager servers are incorrect, or that the Tivoli Access Manager servers have not started.
- Ensure that the password for the sec_master user is correct.
- Check the `SystemOut.log` file and search for the `AMAS` string to see if any error messages are present.

The server might fail to start after configuring JACC

If the server does not start after JACC is configured, check the following items:

- Ensure that WebSphere Application Server and Tivoli Access Manager use the same Lightweight Directory Access Protocol (LDAP) server.
- If the message “Policy Director Authentication failed″ is displayed, ensure that the:
  - WebSphere Application Server LDAP server ID is the same as the “Administrator user” in the Tivoli Access Manager JACC configuration panel.
  - Verify that the Tivoli Access Manager Administrator distinguished name (DN) is correct.
  - Verify that the password of the Tivoli Access Manager administrator has not expired and is valid.
  - Ensure that the account is valid for the Tivoli Access Manager administrator.
- If a message such as `socket can't be opened for` *`xxxx`* (where *xxxx* is a number) is displayed, take the following actions:
  1. Go to the *`profile_root`*`/etc/tam` directory.
  2. Change *xxxx* to an available port number in the `amwas.commomconfig.properties` file, and the `amwas*cellName_dmgr.properties` file if the deployment manager failed to start. If the node failed to start, change *xxx* to an available port number in the `amwas*cellName_nodeName_.properties` file. If the Application Server failed to start, change *xxxx* in the `amwas*cellname_nodeName_serverName.properties` file.

The application might not deploy properly

When you click **Save**, the policy and role information is propagated to the Tivoli Access Manager policy. This process might take some time to finish. If the save fails, you must uninstall the application and then reinstall it.

To access an application after it is installed, you must wait 30 seconds, by default, to start the application after you save.

The startServer command might fail after you configure Tivoli Access Manager or a clean uninstall did not take place after unconfiguring JACC.

If the cleanup for JACC unconfiguration or start server fails after JACC is configured, take the following actions:

- Remove Tivoli Access Manager properties files from WebSphere Application Server. For each application server in a Network Deployment (ND) environment with N servers defined (for example, server1, server2).

  The following files must be removed.

  ```
  profile_root/etc/pd/PolicyDirector/PDPerm.properties
  profile_root/etc/pd/PolicyDirector/PdPerm.ks
  profile_root/etc/tam/*
  ```

- Use a utility to clear the security configuration and return the system to the state it was in before you configure the JACC provider for Tivoli Access Manager. The utility removes all of the PDLoginModuleWrapper entries as well as the Tivoli Access Manager authorization table entry from the `security.xml` file, effectively removing the JACC provider for Tivoli Access Manager. Backup the `security.xml` file before running this utility.

  Enter the following commands:

```
java -Djava.version=1.5 -classpath
"app_server_root/lib/AMJACCProvider.jar:CLASSPATH"
com.tivoli.pd.as.jacc.cfg.CleanSecXML fully_qualified_path/security.xml
```

An ″HPDIA0202w An unknown user name was presented to Access Manager″ error might occur

You might encounter the following error message if you try to use an existing user in a Local Directory Access Protocol (LDAP) user registry with Tivoli Access Manager:

```
AWXJR0008E   Failed to create a PDPrincipal for principal mgr1.:
AWXJR0007E   A Tivoli Access Manager exception was caught. Details are:
"HPDIA0202W  An unknown user name was presented to Access Manager."
```

This problem might be caused by the host name exceeding predefined limits with Tivoli Access Manager when it is configured against MS Active Directory. In WebSphere Application Server, the maximum length of the host name can not exceed 46 characters.

Check that the host name is not fully qualified. Configure the machine so that the host name does not include the host domain.

To correct this error, complete the following steps:

1. On the command line, type the following information to get a Tivoli Access Manager command prompt:

   ```
   pdadmin -a administrator_name -p administrator_password
   ```

   The pdadmin *administrator_name* prompt is displayed. For example:

   ```
   pdadmin -a administrator1 -p passw0rd
   ```

2. At the pdadmin command prompt, import the user from the LDAP user registry to Tivoli Access Manager by typing the following information:

```
user import user_name cn=user_name,o=organization_name,c=country
```

   For example:

   ```
   user import jstar cn=jstar,o=ibm,c=us
   ```

After importing the user to Tivoli Access Manager, you must use the **user modify** command to set the user account to `valid`. The following syntax shows how to use this command:

```
user modify user_name account-valid yes
```

For example:

```
user modify jstar account-valid yes
```

For information on how to import a group from LDAP to Tivoli Access Manager, see the Tivoli Access Manager documentation.

An ″HPDAC0778E The specified user’s account is set to invalid″ error might occur

You might encounter the following error message after you import a user to Tivoli Access Manager and restart the client:

```
AWXJR0008E    Failed to create a PDPrincipal for principal mgr1.:
AWXJR0007E    A Tivoli Access Manager exception was caught.
Details are: "HPDAC0778E   The specified user's account is set to invalid."
```

To correct this error, use the `user modify` command to set the user account to valid. The following syntax shows how to use this command:

```
user modify user_name account-valid yes
```

For example:

```
user modify jstar account-valid yes
```

An ″HPDJA0506E Invalid argument: Null or zero-length user name field for the ACL entry″ error might occur

You might encounter an error similar to the following message when you propagate the security policy information from the application to the provider using the wsadmin **propagatePolicyToJACCProvider** command:

```
AWXJR0035E   An error occurred while  attempting to add member,
                cn=agent3,o=ibm,c=us, to role AgentRole
HPDJA0506E   Invalid argument: Null or zero-length user name field for
                the ACL entry
```

To correct this error, create or import the user, that is mapped to the security role to the Tivoli Access Manager. For more information on propagating the security policy information, see the documentation for your authorization provider.

An WASX7017E: Exception received while running file ″InsuranceServicesSingle.jacl″ error might occur

After the JACC provider and Tivoli Access Manager are enabled, when attempting to install the application, which is configured with security roles using the wsadmin command, the following error might occur:

```
WASX7017E: Exception received while running file "InsuranceServicesSingle.jacl";
exception information: com.ibm.ws.scripting.ScriptingException: WASX7111E:
Cannot find a match for supplied option:
"[RuleManager, , , cn=mgr3,o=ibm,c=us|cn=agent3,o=ibm,c=us, cn=ManagerGro
up,o=ibm,c=us|cn=AgentGroup,o=ibm,c=us]" for task "MapRolesToUsers
```

The $AdminApp MapRolesToUsers task option is no longer valid when Tivoli Access Manager is used as the authorization server. To correct the error, change MapRolesToUsers to TAMMapRolesToUsers.

Access denied exceptions accessing applications when using JACC

In the case of Tivoli Access Manager, you might see the following error message.

```
AWXJR0044E: The access decision for Permission, {0}, was denied because either the
PolicyConfiguration or RoleConfiguration objects did not get created successfully at
application installation time. RoleConfiguration exists = {false}, PolicyConfiguration
exists = {false}."
```

If the access denied exceptions are not expected for the application, check the SystemOut.log files to see if the security policy information was correctly propagated to the provider.

If the security policy information for the application is successfully propagated to the provider, the audit statements with the message key SECJ0415I appear. However, if there was a problem propagating the security policy information to the provider (for example: network problems, JACC provider is not available), the SystemOut.log files contain the error message with the message keys SECJ0396E (during install) or SECJ0398E (during modification). The installation of the application is not stopped due to a failure to propagate the security policy to the JACC provider. Also, in the case of failure, no exception or error messages appear during the save operation. When the problem causing this failure is fixed, run the propagatePolicyToJaccProvider tool to propagate the security policy information to the provider without reinstalling the application. For more information about this task, see the Propagating security policy of installed applications to a JACC provider using wsadmin scripting topic in the *Securing applications and their environment* PDF book.

An ″HPDBA0219E: An error occurred reading data from an SSL connection″ might occur

An error message (HPDBA0219E) might appear in dmgr SystemOut.log when you install an application on WebSphere Application Server for Network Deployment (ND) and a managed node with Tivoli Access Manager is enabled.

If the error occurs, then the security policy data of recently deployed applications might not be immediately available. The policy data is available based on the server replicate time of the Tivoli Access Manager. This is defaulted to 30 seconds after all updates have been completed. To ensure that the latest policy data is available, log on to the pdadmin console and type: `server replicate`.

Password decoding troubleshooting tips

If the password encoding is corrupted and you cannot decode a password, you can complete one of the following tasks.

- If the password is contained in a server configuration file, edit the file and set the password to the clear text value. After changing the value, restart the server.
- If the password is contained in the `sas.client.props` file or the `soap.client.props` file, edit the file and set the password to the appropriate clear text value. After changing the value, use the PropFilePasswordEncoder utility to encode the password. For more information on the PropFilePasswordEncoder utility, see thePropFilePasswordEncoder command reference.

The profile-specific setupCmdLine QShell script contains a property that you can use to obtain trace information when using the OS400 algorithm with Java clients and administrative commands for WebSphere Application Server. To obtain the trace, set the os400.security.password.debug property to `true`. The trace is printed to standard output.

SPNEGO trust association interceptor (TAI) troubleshooting tips (deprecated)

Presented here is a list of trouble shooting tips useful in diagnosing Simple and Protected GSS-API Negotiation (SPNEGO) TAI problems and exceptions.

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO Web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

The IBM Java Generic Security Service (JGSS) and IBM Simple and Protected GSS-API Negotiation (SPNEGO) providers use a Java virtual machine (JVM) custom property to control trace information. The SPNEGO TAI uses the JRas facility to allow an administrator to trace only specific classes. The following important trace specifications or JVM custom properties should be used to debug the TAI using tracing.

Table 2. SPNEGO TAI trace specifications

Trace	Use
com.ibm.security.jgss.debug	Set this **JVM Custom Property** to `all` to trace through JGSS code. Messages appear in the `trace.log` file, and **SystemOut.log**.
com.ibm.security.krb5.Krb5Debug	Set this **JVM Custom Property** to `all` to trace through the Kerberos5-specific JGSS code. Messages appear in the `trace.log` file, and **SystemOut.log**.
*com.ibm.ws.security.spnego.**	Set this trace on using the **administrative console > troubleshooting > Logging and Tracing > server1 > Change Log Detail Levels > com.ibm.ws.security.spnego.***. Messages appear in the `trace.log` file.

Problem: WebSphere Application Server and the Active Directory (AD) Domain Controller’s time are not synchronized within 5 minutes.

Symptom

```
[2/24/06 13:12:46:093 CST] 00000060 Context       2 com.ibm.ws.security.spnego.Context
    begin GSSContext accepted
[2/24/06 13:12:46:093 CST] 00000060 Context       E com.ibm.ws.security.spnego.Context
    begin
CWSPN0011E: An invalid SPNEGO token has been encountered while authenticating a
    HttpServletRequest:
0000:  60820160 06062b06 01050502 a1820154    `..` ..+. .... ...T
0010:  30820150 a0030a01 01a10b06 092a8648    0..P .... .... .*.H
0020:  82f71201 0202a282 013a0482 01366082    .... .... .:.. .6`.
0030:  01320609 2a864886 f7120102 0203007e    .2.. *.H. .... ...~
0040:  82012130 82011da0 03020105 a1030201    ..!0 .... .... ....
0050:  1ea41118 0f323030 36303232 34313931    .... .200 6022 4191
0060:  3234365a a5050203 016b48a6 03020125    246Z .... .kH. ...%
0070:  a9161b14 57535345 432e4155 5354494e    .... WSSE C.AU STIN
0080:  2e49424d 2e434f4d aa2d302b a0030201    .IBM .COM .-0+ ....
0090:  00a12430 221b0448 5454501b 1a773230    ..$0 "..H TTP. .w20
00a0:  30337365 63646576 2e617573 74696e2e    03se cdev .aus tin.
00b0:  69626d2e 636f6dab 81aa1b81 a76f7267    ibm. com. .... .org
00c0:  2e696574 662e6a67 73732e47 53534578    .iet f.jg ss.G SSEx
00d0:  63657074 696f6e2c 206d616a 6f722063    cept ion,  maj or c
00e0:  6f64653a 2031302c 206d696e 6f722063    ode:  10,  min or c
00f0:  6f64653a 2033370a 096d616a 6f722073    ode:  37. .maj or s
0100:  7472696e 673a2044 65666563 74697665    trin g: D efec tive
0110:  20746f6b 656e0a09 6d696e6f 72207374     tok en.. mino r st
0120:  72696e67 3a20436c 69656e74 2074696d    ring : Cl ient  tim
0130:  65204672 69646179 2c204665 62727561    e Fr iday , Fe brua
0140:  72792032 342c2032 30303620 61742031    ry 2 4, 2 006  at 1
0150:  3a31323a 34352050 4d20746f 6f20736b    :12: 45 P M to o sk
0160:  65776564                               ewed
```

User Action

The preferred way to resolve this issue is to synchronize the WebSphere Application Server system time to within 5 minutes of the AD server’s time. A best practice is to use a time server to keep all systems synchronized. You can also add or adjust the clockskew parameter in the Kerberos configuration file.

Note: The default for the clockskew parameter is 300 seconds (or 5 minutes).

Problem: No factory available to create a name for mechanism 1.3.6.1.5.5.2.

Problem

Getting an exception: No factory available to create a name for mechanism 1.3.6.1.5.5.2. There is no factory available to process the creation of a name for the specific mechanism.

Symptom

```
[4/8/05 22:51:24:542 EDT] 5003e481 SystemOut     O [JGSS_DBG_PROV] Provider
        IBMJGSSProvider version 1.01 does not support mech 1.3.6.1.5.5.2
[4/8/05 22:51:24:582 EDT] 5003e481 ServerCredent >
        com.ibm.ws.security.spnego.ServerCredential initialize ENTRY
SPNEGO014: Kerberos initialization Failure: org.ietf.jgss.GSSException, major code: 2,
        minor code: 0
 major string: Unsupported mechanism
 minor string: No factory available to create name for mechanism 1.3.6.1.5.5.2
 at com.ibm.security.jgss.i18n.I18NException.throwGSSException
        (I18NException.java:30)
 at com.ibm.security.jgss.GSSManagerImpl.a(GSSManagerImpl.java:36)
 at com.ibm.security.jgss.GSSCredentialImpl.add(GSSCredentialImpl.java:217)
 at com.ibm.security.jgss.GSSCredentialImpl.<init>(GSSCredentialImpl.java:264)
```

User Action

Check the `java.security` file to ensure it contains the IBMSPNEGO security provider and that the provider is defined correctly. The `java.security` file should contain a line similar to:

```
security.provider.6=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
```

Problem: Getting an exception as the JGSS library is trying to process the SPNEGO token.

Symptom

The following error is displayed as the JGSS library is trying to process the SPNEGO token.

```
Error authenticating request. Reporting to client
Major code = 11, Minor code = 31
org.ietf.jgss.GSSException, major code: 11, minor code: 31
 major string: General failure, unspecified at GSSAPI level
 minor string: Kerberos error while decoding and verifying token:
               com.ibm.security.krb5.internal.KrbException, status code: 31
 message: Integrity check on decrypted field failed
```

Description

This exception is the result of encoding the ticket using one key and attempting to decode it using a different key. There are number of possible reasons for this condition:

1. The Kerberos keytab file has not been copied to the server machine after it has been regenerated.
2. The Kerberos configuration points to the wrong Kerberos keytab file.
3. The Kerberos service principal name (SPN) has been defined to the Active Directory more than once. You have another userid defined with the same SPN or defined with the same SPN with a port defined also. The following example demonstrates how this condition can occur:

   SAME SPN but different user ids

   ```
   setspn -a HTTP/myHost.austin.ibm.com user1
    setspn -a HTTP/myHost.austin.ibm.com user2
   ```

   SAME SPN and same user ids, one without a port number, one with a port number

   ```
   setspn -a HTTP/myHost.austin.ibm.com user
    setspn -a HTTP/myHost.austin.ibm.com:9080 user
   ```

User Action

If the problem is with the Kerberos keytab file, then regenerate the keytab file. If the problem is with multiple SPN definitions, then remove the extra or conflicting SPN, confirm that the SPN is no longer registered with the Active Directory, and then add the SPN. The Active Directory may need to be searched for other entries with SPNs defined that clash with the SPN.

To confirm that the SPN is not registered, the command:

```
setspn –l userid
```

should return with the following response:

```
Cannot find account userid
```

Problem: Single sign-on is not occurring.

Symptom

When tracing is enabled, the following message appears:

```
[2/27/06 14:28:04:191 CST] 00000059 SpnegoHandler <
        com.ibm.ws.security.spnego.SpnegoHandler handleRequest: Received a
        non-SPNEGO Authorization Header RETURN
```

Description

The client is returning an NT LAN manager (NTLM) response to the authorize challenge, not a SPNEGO token. This condition can be occur due to any of the following reasons:

- The client has not been configured properly.
- The client is not using a supported browser. For example, when using Microsoft Internet Explorer 5.5, SP1 responds with a non-SPNEGO authentication header.
- The user has not logged into the Active Directory domain, or into a trusted domain, or the client used does not support integrated authentication with Windows – in this case, the SPNEGO TAI is working properly.
- The user is accessing a service defined on the same machine upon which the client is running (local host). Microsoft Internet Explorer resolves the host name of the URL to `http://localhostsomeURL` instead of a fully qualified name.
- The SPN is not found in the Active Directory. The SPN must be of the format `HTTP/server.realm.com`. The command to add the SPN is

  ```
  setspn –a HTTP/server.realm.com userid
  ```
- The Kerberos service principal name (SPN) has been defined to the Active Directory more than once. You have either another user ID defined with the same SPN or another userid defined with the same SPN with a port number defined. The following categories describe these conditions:

  Same SPN but with differing user IDs

  – `setspn -a HTTP/myappserver.austin.ibm.com user1`

  – `setspn -a HTTP/myappserver.austin.ibm.com user2`

  Same SPN and same user IDs, one with a port number defined

  – `setspn -a HTTP/myappserver.austin.ibm.com user3`

  – `setspn -a HTTP/myappserver.austin.ibm.com:9080 user3`

User Action

If the SPN is defined incorrectly as `HTTP/server.realm.com@REALM.COM` with the addition of `@REALM.COM`, then delete the user, redefine the user, and redefine the SPN.

If the problem is with the Kerberos keytab file, then regenerate the keytab file.

If the problem is with either category of multiple SPN definitions, then remove the extra or conflicting SPN, confirm that the SPN is no longer registered with the Active Directory, and then add the SPN. You can search the Active Directory for other SPN entries that are causing multiple SPN definitions. The following commands are useful to determine multiple SPN definitions:

`setspn ?L userid`
: Returns the message, `cannot find account userid`, if the SPN is not registered.

`setspn -L`
: Displays the SPNs that exist.

Problem: Credential Delegation is not working.

Symptom

An invalid option is detected. When tracing is enabled, the following message is displayed:

```
com.ibm.security.krb5.KrbException, status code: 101 message: Invalid option in
ticket request
```

Description

The Kerberos configuration file is not properly configured.

User Action

Ensure that neither renewable, nor proxiable are set to true.

Problem: Unable to get SSO working using RC4-HMAC encryption.

Symptom

Examine the following message in the trace that you receive when trace is turned on:

```
com.ibm.security.krb5.internal.crypto.KrbCryptoException, status code: 0
 message: Checksum error; received checksum does not match computed checksum
```

Description

RC4-HMAC encryption is not supported with a Microsoft Windows 2000 Kerberos key distribution center (KDC). To confirm this condition, examine the trace and identify where the exception is thrown. The content of the incoming ticket should be visible in the trace. Although the incoming ticket is encrypted, the SPN for the service is readable. If a Microsoft Windows 2000 KDC is used and the system is configured to use RC4-HMAC, the string representing the ticket for `userid@REALM` (instead of the expected `HTTP/hostname.realm@REALM`) is displayed. For example, this is beginning of the ticket received from a Microsoft Windows 2000 KDC:

```
0000: 01 00 6e 82 04 7f 30 82  04 7b a0 03 02 01 05 a1  ..n...0.........
0010: 03 02 01 0e a2 07 03 05  00 20 00 00 00 a3 82 03  ................
0020: a5 61 82 03 a1 30 82 03  9d a0 03 02 01 05 a1 0a  .a...0..........
0030: 1b 08 45 50 46 44 2e 4e  45 54 a2 18 30 16 a0 03  ...REALM.COM.0..
0040: 02 01 01 a1 0f 30 0d 1b  0b 65 70 66 64 77 61 73  .....0...userid
0050: 75 6e 69 74 a3 82 03 6e  30 82 03 6a a0 03 02 01  .a.f...n0..j....
```

The realm is REALM.COM. The service name is userid. A correctly formed ticket for the same SPN is:

```
0000: 01 00 6e 82 04 56 30 82  04 52 a0 03 02 01 05 a1  ..n..V0..R......
0010: 03 02 01 0e a2 07 03 05  00 20 00 00 00 a3 82 03  ................
0020: 82 61 82 03 7e 30 82 03  7a a0 03 02 01 05 a1 0a  .a...0..z.......
0030: 1b 08 45 50 46 44 2e 4e  45 54 a2 2a 30 28 a0 03  ..REALM.COM.0...
0040: 02 01 02 a1 21 30 1f 1b  04 48 54 54 50 1b 17 75  .....0...HTTP..u
0050: 73 31 30 6b 65 70 66 77  61 73 73 30 31 2e 65 70  serid.realm.com.
0060: 66 64 2e 6e 65 74 a3 82  03 39 30 82 03 35 a0 03  ...n.....90..5..
```

User Action

To correct the problem, either use the Single data encryption standard (DES) or use a Microsoft Windows 2003 Server for a KDC. Remember to regenerate the SPN, and the Kerberos keytab file.

Problem: User receives the following message when accessing a protected URL through the SPNEGO SSO.

Symptom

Examine the following message:

```
Bad Request

Your browser sent a request that this server could not understand.
Size of request header field exceeds server limit.

Authorization: Negotiate YII......
```

Description

This message is generated by the Apache/IBM HTTP Server. This server is indicating that the authorization header returned by the user's browser is too large. The long string that follows the word Negotiate (in the error message above) is the SPNEGO token. This SPNEGO token is a wrapper of the Microsoft Windows Kerberos token. Microsoft Windows includes the user's PAC information in the Kerberos token. The more security groups that the user belongs to, the more PAC information is inserted in the Kerberos token, and the larger the SPNEGO becomes. IBM HTTP Server 2.0 (also Apache 2.0 and IBM HTTP Server 6.0) limit the size of any acceptable HTTP header to be 8K. In Microsoft Windows domains having many groups, and with user membership in many groups, the size of the user's SPNEGO token may exceed the 8K limit.

User Action

If possible, reduce the number of security groups the user is a member of. IBM HTTP Server 2.0.47 cumulative fix PK01070 allows for HTTP header sizes up to and beyond the Microsoft limit of 12K. WebSphere Application Server Version 6.0 users can obtain this fix in fixpack 6.0.0.2.

Note: Non-Apache based Web servers may require differing solutions.

Problem: Even with JGSS tracing disabled, some KRB_DBG_KDC messages appear in the SystemOut.log.

Symptom

Examine the SystemOut.log and note the some KRB_DBG_KDC messages appear there even with JGSS tracing disabled.

Description

While most of the JGSS tracing is controlled by the com.ibm.security.jgss.debug property, a small set of messages are controlled by the com.ibm.security.krb5.Krb5Debug property. The com.ibm.security.krb5.Krb5Debug property has a default value to put some messages to the **SystemOut.log**

User Action

.To remove all KRB_DBG_KDC messages from the **SystemOut.log**, set the JVM property as follows:

```
-Dcom.ibm.security.krb5.Krb5Debug=none
```

Problem: When an application contains a custom HTTP 401 error page, the SPNEGO TAI-generated HTTP response page is not displayed in a browser.

Symptom

When an application contains a custom HTTP 401 error page, the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI)-generated HTTP response page is not displayed in a browser.

Description

When an application contains a custom HTTP 401 error page, the SPNEGO TAI-generated HTTP response page is not displayed in a browser. The custom HTTP 401 error page is displayed instead.

User Action

You can customize your HTTP 401 page to include information concerning how to configure your browser to use SPNEGO. For more information, see Configuring the client browser to use SPNEGO TAI (deprecated) and SPNEGO TAI custom properties configuration (deprecated).

Problem: HTTP Post parameters are lost during interaction with the SPNEGO TAI, when stepping down to userid/password login.

Symptom

Note that HTTP Post parameters are lost during interaction with the SPNEGO TAI, when stepping down to userid/password login.

″Stepping down to userid/password login″ means that the Microsoft Internet Explorer tries to respond initially with a SPNEGO token. If this response is unsuccessful, then the Microsoft Internet Explorer tries to respond with a NTLM token that is obtained through a userid/password challenge.

Description

The Microsoft Internet Explorer maintains state during a user's request. If a request was given the response of an ″HTTP 401 Authenticate Negotiate″, and the browser responds with a NTLM token obtained through a userid/password challenge, the browser resubmits the request. If this second request is given a response of an HTML page containing a redirection to the same URL but with new arguments (via Javascript) then the browser does not resubmit the POST parameters.

Note: To avoid this problem, it is critical to NOT perform the automatic redirection. If the user clicks on a link, the problem does not occur.

User Action

The browser responds to the Authenticate/Negotiate challenge with an NTLM token, not an SPNEGO token. The SPNEGO TAI sees the NTLM, and returns back a HTTP 403 response, along with the HTML page. When the browser runs the Javascript redirTimer function, any POST of GET parameters that were present on the original request are lost.

By leveraging the SPN<id>.NTLMTokenReceivedPage property, an appropriate message page can be returned to the user. The default message that is returned (in the absence of a user defined property) is:

```
"<html><head><title>An NTLM Token was Received.</title></head>"
 + "<body>Your browser configuration is correct, but you have not logged into
     a supported Windows Domain."
 + "<p>Please login to the application using the normal login page.</html>";
```

Using the SPN<id>.NTLMTokenReceivedPage property, you can customize the exact response. It is critical that the returned HTML not perform a redirection.

When the SPNEGO TAI has been configured to use the shipped default HTTPHeaderFilter class as the SPN<id>.filterClass, then the SPN<id>.filter can be used to allow the second request to flow directly to the normal WebSphere Application Server security mechanism. In this way, the user experiences the normal authentication mechanism.

An example of such a configuration follows showing the required SPNEGO TAI properties necessary and the HTML file content.

```
****** SPNEGO TAI Property Name ******                 ****** HTML File Content ******
com.ibm.ws.security.spnego.SPN1.hostName               server.wasteched30.torolab.ibm.com
com.ibm.ws.security.spnego.SPN1.filterClass            com.ibm.ws.security.spnego.HTTPHeaderFilter
com.ibm.ws.security.spnego.SPN1.filter                 request-url!=noSPNEGO
com.ibm.ws.security.spnego.SPN1.NTLMTokenReceivedPage  File:///C:/temp/NTLM.html
```

Note: Observe that the filter property instructs the SPNEGO TAI to NOT intercept any HTTP request that contains the string “noSPNEGO”.

Here is an example of a generating a helpful response.

```
<html>
<head>
<title>NTLM Authentication Received </title>
<script language="javascript">
 var purl=""+document.location;
if (purl.indexOf("noSPNEGO")<0) {
  if(purl.indexOf('?')>=0) purl+="&noSPNEGO";
  else purl+="?noSPNEGO";
 }
</script>
</head>
<body>
<p>An NTLM token was retrieved in response to the SPNEGO challenge. It is likely that
you are not logged into a Windows domain.<br>
Click on the following link to get the requested website.
<script language="javascript">
 document.write("<a href='"+purl+"'>");
 document.write("Open the same page using the normal authentication
  mechanism.");
 document.write("</a><br>");
</script>
You will not automatically be redirected.
</body>
</html>
```

Chapter 8. Naming and directory

Troubleshooting namespace problems

When developing or running applications, you might encounter namespace problems.

About this task

Many naming problems can be avoided by fully understanding the key underlying concepts of the Naming service.

1. Review the key concepts of the Naming service, especially the sections about Namespace logical view and Lookup names support in deployment descriptors and thin clients.
2. Review the programming examples that are included in the sections explaining the Java Naming and Directory Interface (JNDI) and CosNaming interfaces.
3. Read "Naming service troubleshooting tips" for additional general information.
4. Read "Application access problems" on page 130 for information on lookup errors.

Naming service troubleshooting tips

Naming is a Java Platform, Enterprise Edition (Java EE) service which publishes and provides access to resources such as connection pools, enterprise beans, and message listeners to client processes. If you have problems in accessing a resource which otherwise appears to be healthy, the naming service might be involved.

To investigate problems with the WebSphere Application Server Naming service:

- Browse the Java virtual machine (JVM) logs for the server which is hosting the resource you are trying to access. Messages starting with `NMSV` are related to the Naming service.
- With WebSphere Application Server running, run the dumpNameSpace tool and pipe, redirect, or ″more″ the output so that it is easily viewed. Running the tool results in a display of objects in the WebSphere Application Server namespace, including the directory path and object name.

  Note: The dumpNameSpace tool does not dump all of the objects in the distributed namespace. It only dumps the objects that are in the local namespace of the process against which the command was run.
- If the object a client needs to access does not appear, use the administrative console to verify that:
  - The server hosting the target resource is started.
  - The Web module or EJB container, if applicable, hosting the target resource is running.
  - The Java Naming and Directory Interface (JNDI) name of the target resource is correct and updated.
  - If the problem resource is remote, that is, not on the same node as the Name Server node, that the JNDI name is fully qualified, including the host name.

    This especially applies to multiple-server configurations.
- View detailed information on the runtime behavior of the Naming service by enabling trace on the following components and reviewing the output:
  - com.ibm.ws.naming.*
  - com.ibm.websphere.naming.*
- If you see an exception that appears to be CORBA related (″CORBA″ appears as part of the exception name) look for a naming-services-specific CORBA minor code, further down in the exception stack, for information on the real cause of the problem. For a list of naming service exceptions and explanations, see the class com.ibm.websphere.naming.WsnCorbaMinorCodes in the API documentation that is included in the Reference section of the information center.

If none of these steps solve the problem:

- For specific problems that can cause access to named object hosted in WebSphere Application Server to fail, see Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client.
- Check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.
- If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Application access problems

To resolve problems encountered when a servlet, JavaServer Pages file, standalone application or other client attempts to access an enterprise bean, ConnectionPool, or other named object hosted by WebSphere Application Server, you must first verify that the target server can be accessed from the client.

Follow these steps:

- From a command prompt on the client's server, enter `ping` *`server_name`* and verify connectivity.
- Use the administrative console to verify that the target resource's application server and, if applicable, enterprise bean (EJB) module or Web module, is started.

Continue only if there is no problem with connectivity and the target resource appears to be running.

What kind of error are you seeing?

- "NameNotFoundException from JNDI lookup operation "
- "CannotInstantiateObjectException from JNDI lookup operation " on page 132
- "Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred " on page 132
- "OperationNotSupportedException from JNDI Context operation" on page 132
- "WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound " on page 132
- "ConfigurationException from ″new InitialContext″ operation or from a JNDI Context operation with a URL name" on page 133
- "ServiceUnavailableException from ″new InitialContext″ operation" on page 133
- "CommunicationException thrown from a ″new InitialContext″ operation" on page 133
- NMSV0605E: A Reference object looked up from the context. See the *Developing and deploying applications* PDF for more information.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

NameNotFoundException from JNDI lookup operation

There are three causes for a NameNotFoundException:

- The lookup name is incorrect.
- The object being looked up is not bound.
- Two servers with the same name running on the same host are being used to interoperate.

Incorrect lookup name

If you encounter a NameNotFoundException when trying to access an enterprise bean, data source, messaging resource, or other resource:

1. Determine the cause of the NameNotFoundException.

Browse the properties of the target object in the administrative console, and verify that the Java Naming and Directory Interface (JNDI) name it specifies matches the JNDI name the client is using.

If you are looking up an object that resides on a server different from the one from which the initial context was obtained, you must use the fully qualified name.

- If access is from another server object such as a servlet accessing an enterprise bean and you are using the default context, not specifying the fully qualified JNDI name, you might get a NameNotFoundException if the object is hosted on a different server.
- If access is from a standalone client, it might be that the object you are attempting access is on a server different from the server from which you obtained the initial context.

2. Use the fully-qualified JNDI name to correct the problem.

   If the object is in a single server, the fully-qualified JNDI name is as follows:

   ```
   cell/nodes/nodeName/servers/serverName/jndiName
   ```

   Note: Objects are not supported in this release.

   If the object is on a server cluster, the fully-qualified JNDI name is as follows:

   ```
   cell/clusters/clusterName/jndiName
   ```

Object being looked up is not bound

To correct a NameNotFoundException where the object being looked up is not bound:

1. If the object being looked up is an application object such as an enterprise bean, ensure that the application is running.
2. Run the dumpNameSpace tool to view the contents of the name space to verify that the object being looked up is bound to the name space with the expected name.

Two servers with the same name running on the same host are being used to interoperate

If an application running on a server in node1 uses a remote object reference to an object that resides on a similarly named server in node2 and both nodes are installed on the same host, several different failures might occur:

- JNDI lookups fail with a NameNotFoundException.
- Object references obtained other than through JNDI lookups fail, most likely with an org.omg.CORBA.OBJECT_NOT_EXIST exception.
- A remote object reference resolves incorrectly to an object in the local process because the object also exists in the local process. That is, a reference to a remote object on the server process in node2 resolves incorrectly to the same kind of object in the local process, which is the server process in node1.

Object references between servers in different nodes and on different hosts do not result in an exception even if the server names are not unique.

To fix the failures, set a Java virtual machine (JVM) custom property for the Object Request Broker (ORB), com.ibm.websphere.orb.uniqueServerName, to `true` for either or both servers:

1. In the administrative console, click **Servers** → **Server Types** → **WebSphere application servers** → ***server_name*** → **Java and Process Management** → **Process definition** → **Java virtual machine** → **Custom properties** → **New**.
2. On the Custom Properties settings page, define the custom property:
   a. For **Name**, specify `com.ibm.websphere.orb.uniqueServerName`.
   b. For **Value**, specify `true`.
3. Click **OK**.
4. Click **Save** on the console task bar.
5. Restart the application server.

To prevent similar failures with the node agent, you can set the node agent Java virtual machine custom property ORB, com.ibm.websphere.orb.uniqueServerName, to `true`.

CannotInstantiateObjectException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource, possible causes include:

- A serialized Java object is being looked up, but the necessary classes required to deserialize it are not in the runtime environment.
- A Reference object is being looked up, and the associated factory used to process it as part of the lookup processing is failing.

To determine the precise cause of the problem:

- Look at the relevant logs for exceptions immediately preceding the CannotInstantiateObjectException. If it is a java.lang.NoClassDefFoundError or java.lang.ClassNotFoundException, make sure the class referenced in the error message can be located by the class loader. See the section Class loading in the *Developing and deploying applications* PDF book.

  View the JVM logs.
- Print out the stack trace for the root cause and look for the factory class. It will be called by `javax.naming.NamingManager.getObjectInstance()`. The reason for the failure will depend on the factory implementation, and may require you to contact the developer of the factory class.

Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred

This error is informational only and is provided in case the exception is related to an actual problem. Most of the time, it is not. If it is, the log file should contain adjacent entries to provide context.

- If no problems are being experienced, ignore this message. Also ignore the message if the problem you are experiencing does not appear to be related to the exception being reported and if there are no other adjacent error messages in the log.
- If a problem is being experienced, look in the log for underlying error messages.
- The information provided in message NMSV0610I can provide valuable debug data for other adjacent error messages posted in response to the Naming exception that occurred.

OperationNotSupportedException from JNDI Context operation

This error has two possible causes:

- An update operation, such as a bind, is being performed with a name that starts with ″`java:comp/env`″. This context and its subcontexts are read-only contexts.
- A Context bind or rebind operation of a non-CORBA object is being performed on a remote name space that does not belong to the product. Only CORBA objects can be bound to these CosNaming name spaces.

To determine which of these errors is causing the problem, check the full exception message.

WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound

This error occurs two enterprise bean server applications were installed on the same server such that a binding name conflict occurred. That is, a jndiName value is the same in the two applications' deployment descriptors. The error will surface during server startup when the second application using that jndiName value is started.

To verify that this is the problem, examine the deployment descriptors for all enterprise bean server applications running in the server in search for a jndiName that is specified in more than one enterprise bean application.

To correct the problem, change any duplicate jndiName values to ensure that each enterprise bean in the server process is bound with a different name.

ConfigurationException from ″new InitialContext″ operation or from a JNDI Context operation with a URL name

If you are attempting to obtain an initial JNDI context, a configuration exception can occur because an invalid JNDI property value was passed to the InitialContext constructor. This includes JNDI properties set in the System properties or in some `jndi.properties` file visible to the class loader in effect. A malformed provider URL is the most likely property to be incorrect. If the JNDI client is being run as a thin client such that the `CLASSPATH` is set to include all of the individual jar files required, make sure the `.jar` file containing the properties file `com/ibm/websphere/naming/jndiprovider.properties` is in the `CLASSPATH`.

If the exception is occurring from a JNDI Context call with a name in the form of a URL, the current JNDI configuration may not be set up properly so that the required factory class name cannot be determined, or the factory may not be visible to the class loader currently in effect. If the name is a Java: URL, the JNDI client must be running in a Java Platform, Enterprise Edition (Java EE) client or server environment. That is, the client must be running in a container.

Check the exception message to verify the cause.

If the exception is being thrown from the InitialContext constructor, correct the property setting or the `CLASSPATH`.

If the exception is being thrown from a JNDI Context method, make sure the property `java.naming.factory.url.pkgs` includes the package name for the factory required for the URL scheme in the name. URL names with the Java scheme can only be used while running in a container.

ServiceUnavailableException from ″new InitialContext″ operation

This exception indicates that some unexpected problem occurred while attempting to contact the name server to obtain an initial context. You can query the ServiceUnavailableException for a root cause. Check the root cause for more information. Some of the problems described for CommunicationExceptions might also result in a ServiceUnavailableException.

Since this exception is triggered by an unexpected error, there is no probable cause to confirm. If the root cause exception does not indicate what the probable cause is, investigate the possible causes listed for CommunicationExceptions.

CommunicationException thrown from a ″new InitialContext″ operation

The name server identified by the provider URL cannot be contacted to obtain the initial JNDI context. There are many possible causes for this problem, including:

- The host name or port in the provider URL is incorrect.
- The host name cannot be resolved into an IP address by the domain name server, or the IP address does not match the IP address which the server is actually running under.
- A firewall on the client or server is preventing the port specified in the provider URL from being used.

To correct this problem:

- Make sure the provider URL and the network configurations on the client and server machines are correct.
- Make sure the host name can be resolved into an IP address which can be reached by the client machine. You can do this using the ping command.
- If you are running a firewall, make sure that use of the port specified in the provider URL will be allowed.

Viewing a namespace dump

To understand why a naming operation is failing, view the dump of a namespace. You can use the dumpNameSpace tool to dump the contents of a namespace accessed through a name server. The dumpNameSpace tool is based on Java Naming and Directory Interface (JNDI).

Before you begin

Start the naming service.

When you run the dumpNameSpace tool, the naming service must be active. The dumpNameSpace tool cannot dump namespaces local to the server process, such as those with java: and local: URL schemes. The local: namespace contains references to enterprise beans with local interfaces. Use the namespace dump utility for java:, local: and server namespaces to dump java: and local: namespaces.

Decide which parameters to use for the dumpNameSpace tool. “dumpNameSpace tool” on page 136 describes the supported keywords and values.

About this task

The dumpNameSpace tool dumps the server root context for the server at the specified host and port unless you specify a non-default starting context which precludes it. The tool does not dump the server root contexts for other servers.

You can run the tool from a command line or using its program interface.

- Run the tool from a command line.

  Enter the dumpNameSpace command from the *app_server_root*/bin directory.

  ```
  dumpNameSpace [[-keyword value]...]
  ```

  There are several command-line options to choose from. For detailed help on the options, enter either of the following commands:

  ```
  dumpNameSpace -help
  ```

  or:

  ```
  dumpNameSpace -?
  ```

  Invoke the namespace dump tool from a command line by entering either of the following commands:

  ```
  dumpNameSpace -host myhost.mycompany.com -port 901
  ```

  or:

  ```
  dumpNameSpace -url corbaloc:iiop:myhost.mycompany.com:901
  ```

- Run the tool from a Java program.

  You can access the dumpNameSpace tool through its program interface. Refer to the class `com.ibm.websphere.naming.DumpNameSpace` in the WebSphere Application Server API documentation for details on the DumpNameSpace program interface.

  Add statements such as the following to your Java program to invoke the namespace dump tool from a Java program:

```
{
   ...
   import javax.naming.Context;
   import javax.naming.InitialContext;
   import com.ibm.websphere.naming.DumpNameSpace;
   ...
   java.io.PrintStream filePrintStream = ...
   Context ctx = new InitialContext();
   // Starting context for dump
   ctx = (Context) ctx.lookup("cell/nodes/node1/servers/server1");
   DumpNameSpace dumpUtil =
```

```
      new DumpNameSpace(filePrintStream, DumpNameSpace.SHORT);
   dumpUtil.generateDump(ctx);
   ...
}
```

Results

Namespace dump output is sent to the console. It is also written to the file DumpNameSpace.log in the server's log directory.

The tool dumps output in short or long format. Namespace dump output on single-server installations includes only entries for a single server. Output such as the following for a multiple-server installation includes bindings for multiple servers in the cell. This multiple-server output is in the **SHORT** dump format:

```
Getting the initial context
Getting the starting context

==============================================================================
Namespace Dump
   Provider URL: corbaloc:iiop:localhost:9810
   Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
   Requested root context: cell
   Starting context: (top)=outpostNetwork
   Formatting rules: jndi
   Time of dump: Mon Sep 16 18:35:03 CDT 2002
==============================================================================


==============================================================================
Beginning of Namespace Dump
==============================================================================

    1 (top)
    2 (top)/domain                                   javax.naming.Context
    2    Linked to context: outpostNetwork
    3 (top)/cells                                    javax.naming.Context
    4 (top)/clusters                                 javax.naming.Context
    5 (top)/clusters/Cluster1                        javax.naming.Context
    6 (top)/cellname                                 java.lang.String
    7 (top)/cell                                     javax.naming.Context
    7    Linked to context: outpostNetwork
    8 (top)/deploymentManager                        javax.naming.Context
    8    Linked to URL: corbaloc::outpost:9809/NameServiceServerRoot
    9 (top)/nodes                                    javax.naming.Context
   10 (top)/nodes/will2                              javax.naming.Context
   11 (top)/nodes/will2/persistent                   javax.naming.Context
   12 (top)/nodes/will2/persistent/SomeObject        SomeClass
   13 (top)/nodes/will2/nodename                     java.lang.String
   14 (top)/nodes/will2/domain                       javax.naming.Context
   14    Linked to context: outpostNetwork
   15 (top)/nodes/will2/cell                         javax.naming.Context
   15    Linked to context: outpostNetwork
   16 (top)/nodes/will2/servers                      javax.naming.Context
   17 (top)/nodes/will2/servers/server1              javax.naming.Context
   18 (top)/nodes/will2/servers/will2                javax.naming.Context
   19 (top)/nodes/will2/servers/member2              javax.naming.Context
   20 (top)/nodes/will2/node                         javax.naming.Context
   20    Linked to context: outpostNetwork/nodes/will2
   21 (top)/nodes/will2/nodeAgent                    javax.naming.Context
   22 (top)/nodes/outpost                            javax.naming.Context
   23 (top)/nodes/outpost/node                       javax.naming.Context
   23    Linked to context: outpostNetwork/nodes/outpost
   24 (top)/nodes/outpost/nodeAgent                  javax.naming.Context
   24    Linked to URL: corbaloc::outpost:2809/NameServiceServerRoot
   25 (top)/nodes/outpost/persistent                 javax.naming.Context
   26 (top)/nodes/outpost/nodename                   java.lang.String
   27 (top)/nodes/outpost/domain                     javax.naming.Context
   27    Linked to context: outpostNetwork
   28 (top)/nodes/outpost/servers                    javax.naming.Context
   29 (top)/nodes/outpost/servers/server1            javax.naming.Context
   30 (top)/nodes/outpost/servers/server1/url        javax.naming.Context
   31 (top)/nodes/outpost/servers/server1/url/CatalogDAOSQLURL
   31                                                java.net.URL
   32 (top)/nodes/outpost/servers/server1/mail       javax.naming.Context
   33 (top)/nodes/outpost/servers/server1/mail/PlantsByWebSphere
   33                                                javax.mail.Session
   34 (top)/nodes/outpost/servers/server1/TransactionFactory
   34                                                com.ibm.ejs.jts.jts.ControlSet$LocalFactory
   35 (top)/nodes/outpost/servers/server1/servername java.lang.String
   36 (top)/nodes/outpost/servers/server1/WSsamples  javax.naming.Context
   37 (top)/nodes/outpost/servers/server1/WSsamples/TechSampDatasource
   37                                                TechSamp
   38 (top)/nodes/outpost/servers/server1/thisNode   javax.naming.Context
   38    Linked to context: outpostNetwork/nodes/outpost
   39 (top)/nodes/outpost/servers/server1/cell       javax.naming.Context
   39    Linked to context: outpostNetwork
   40 (top)/nodes/outpost/servers/server1/eis        javax.naming.Context
   41 (top)/nodes/outpost/servers/server1/eis/DefaultDatasource_CMP
   41                                                Default_CF
   42 (top)/nodes/outpost/servers/server1/eis/WSsamples  javax.naming.Context
   43 (top)/nodes/outpost/servers/server1/eis/WSsamples/TechSampDatasource_CMP
   43                                                TechSamp_CF
   44 (top)/nodes/outpost/servers/server1/eis/jdbc   javax.naming.Context
   45 (top)/nodes/outpost/servers/server1/eis/jdbc/PlantsByWebSphereDataSource_CMP
   45                                                PLANTSDB_CF
   46 (top)/nodes/outpost/servers/server1/eis/jdbc/petstore
   46                                                javax.naming.Context
   47 (top)/nodes/outpost/servers/server1/eis/jdbc/petstore/PetStoreDB_CMP
   47                                                PetStore_CF
   48 (top)/nodes/outpost/servers/server1/eis/jdbc/CatalogDB_CMP
   48                                                Catalog_CF
   49 (top)/nodes/outpost/servers/server1/jta        javax.naming.Context
   50 (top)/nodes/outpost/servers/server1/jta/usertransaction
   50                                                java.lang.Object
   51 (top)/nodes/outpost/servers/server1/DefaultDatasource
   51                                                Default Datasource
   52 (top)/nodes/outpost/servers/server1/jdbc       javax.naming.Context
   53 (top)/nodes/outpost/servers/server1/jdbc/CatalogDB CatalogDB
   54 (top)/nodes/outpost/servers/server1/jdbc/petstore  javax.naming.Context
   55 (top)/nodes/outpost/servers/server1/jdbc/petstore/PetStoreDB
   55                                                PetStoreDB
   56 (top)/nodes/outpost/servers/server1/jdbc/PlantsByWebSphereDataSource
   56                                                PLANTSDB
   57 (top)/nodes/outpost/servers/outpost            javax.naming.Context
   57    Linked to URL: corbaloc::outpost:2809/NameServiceServerRoot
   58 (top)/nodes/outpost/servers/member1            javax.naming.Context
   59 (top)/nodes/outpost/cell                       javax.naming.Context
   59    Linked to context: outpostNetwork
   60 (top)/nodes/outpostManager                     javax.naming.Context
   61 (top)/nodes/outpostManager/domain              javax.naming.Context
   61    Linked to context: outpostNetwork
   62 (top)/nodes/outpostManager/cell                javax.naming.Context
   62    Linked to context: outpostNetwork
   63 (top)/nodes/outpostManager/servers             javax.naming.Context
   64 (top)/nodes/outpostManager/servers/dmgr        javax.naming.Context
   64    Linked to URL: corbaloc::outpost:9809/NameServiceServerRoot
   65 (top)/nodes/outpostManager/node                javax.naming.Context
   65    Linked to context: outpostNetwork/nodes/outpostManager
   66 (top)/nodes/outpostManager/nodename            java.lang.String
   67 (top)/persistent                               javax.naming.Context
   68 (top)/persistent/cell                          javax.naming.Context
   68    Linked to context: outpostNetwork
   69 (top)/legacyRoot                               javax.naming.Context
   69    Linked to context: outpostNetwork/persistent
   70 (top)/persistent/AnotherObject                 AnotherClass

==============================================================================
End of Namespace Dump
==============================================================================
```

What to do next

If you cannot use the dumpNameSpace command line utility to dump the java: namespace for a Java Platform, Enterprise Edition (Java EE) specification application because the application's java: namespace is accessible only by that application, run the namespace dump utility for java:, local: and server namespaces.

dumpNameSpace tool

You can use the dumpNameSpace tool to dump the contents of a namespace accessed through a name server. The dumpNameSpace tool is based on Java Naming and Directory Interface (JNDI).

When you run the dumpNameSpace tool, the naming service must be active. The dumpNameSpace tool cannot dump namespaces local to the server process, such as those with java: and local: URL schemes. The local: namespace contains references to enterprise beans with local interfaces. Use the namespace dump utility for java:, local: and server namespaces to dump java: and local: namespaces.

The tool dumps the server root context for the server at the specified host and port unless you specify a non-default starting context which precludes it. The tool does not dump the server root contexts for other servers.

Running dumpNameSpace

You can run the tool from a command line or using its program interface. This topic describes command-line invocations. To access the dumpNameSpace tool through its program interface, refer to the class `com.ibm.websphere.naming.DumpNameSpace` in the WebSphere Application Server API documentation.

To run the tool from a command line, enter the `dumpNameSpace` command from the *app_server_root*/bin directory.

```
dumpNameSpace [[-keyword value]...]
```

If you run the dumpNameSpace tool with security enabled and the com.ibm.CORBA.loginSource property is set in the *profile_root*/properties/sas.client.props file, a login prompt is displayed.

If you cancel the login prompt, the dumpNameSpace tool continues outbound with an ″UNAUTHENTICATED″ credential. Thus, by default, an ″UNAUTHENTICATED″ credential is used that is equivalent to the ″Everyone″ access authorization policy. You can modify this default setting by changing the value for the `com.ibm.CSI.performClientAuthenticationRequired` property to `true` in the *app_server_root*/properties/sas.client.props file.

If you do not set the `com.ibm.CORBA.loginSource` property in the sas.client.props file, the dumpNameSpace tool continues outbound with the user name and password that is set in the credential.

If Kerberos (KRB5) is enabled for administrative authentication, the `authenticationTarget` supports both BasicAuth and KRB5. To use Kerberos authentication, you must update the sas.client.props, soap.client.props, and ipc.client.props files according to the connector type. When using Kerberos authentication, the user password does not flow across the wire. A one-way hash of the password identifies the client.

Parameters

The keywords and associated values for the dumpNameSpace tool follow:

-host *myhost.company.com*

Indicates the bootstrap host or the WebSphere Application Server host whose namespace you want to dump. The value defaults to `localhost`. Specify a value for `-host` if the tool is not run from the local machine. The `-host` parameter instructs the tool to connect to a server on a remote machine. For example, run

```
dumpNameSpace -host myhost.mycompany.com
```

to display the namespace of the server running on *myhost.mycompany.com*.

-port *nnn*

Indicates the bootstrap port which, if not specified, defaults to `2809`.

-root { cell | server | node | host | legacy | tree | default }

Indicates the root context to use as the initial context for the dump. The applicable root options and default root context depend on the type of name server from which the dump is being obtained. Descriptions of `-root` options follow.

For WebSphere Application Server Version 5.0 or later servers:

cell	DumpNameSpace default for product Version 5.0 or later servers. Dumps the tree starting at the cell root context.
server	Dumps the tree starting at the server root context.
node	Dumps the tree starting at the node root context.
tree	Dumps the tree starting at the tree root context.

For all WebSphere Application Server and other name servers:

default	Dumps the tree starting at the initial context which JNDI returns by default for that server type. This is the only `-root` option that is compatible with non-product name servers.

-url *some_provider_URL*
Indicates the value for the java.naming.provider.url property used to get the initial JNDI context. This option can be used in place of the `-host`, `-port`, and `-root` options. If the `-url` option is specified, the `-host`, `-port`, and `-root` options are ignored.

-factory *com.ibm.websphere.naming.WsnInitialContextFactory*
Indicates the initial context factory to be used to get the JNDI initial context. The value defaults to *com.ibm.websphere.naming.WsnInitialContextFactory*. The default value generally does not need to be changed.

-startAt *some/subcontext/in/the/tree*
Indicates the path from the bootstrap host's root context to the top level context where the dump should begin. The tool recursively dumps subcontexts below this point. It defaults to an empty string, that is, the bootstrap host root context.

-format { jndi | ins }

jndi	The default. Displays name components as atomic strings.
ins	Shows name components parsed using Interoperable Naming Service (INS) rules (`id.kind`).

-report { short | long }

short	The default. Dumps the binding name and bound object type. This output is also provided by JNDI Context.list().
long	Dumps the binding name, bound object type, local object type, and string representation of the local object (that is, the IORs, string values, and other values that are printed). For objects of user-defined classes to display correctly with the long report option, you might need to add their containing directories to the list of directories searched. Set the environment variable `WAS_USER_DIRS` at a command line. The value can include one or more directories. All .zip, .jar, and .class files in the specified directories can then be resolved by the class loader when running the dumpNameSpace tool.

-traceString *"some.package.name.to.trace.*=all=enabled"*
Represents the trace string with the same format as that generated by the servers. The output is sent to the file DumpNameSpaceTrace.out.

Return codes

The dumpNameSpace tool has the following return codes:

Return code	Description
0	Normal system exit. No error resulted from running dumpNameSpace.
1	Error in getting the starting context
2	Other error occurred with exception. Running dumpNameSpace resulted in an error other than an error in getting the starting context.

Return code	Description
3	Unsupported option specified

Viewing `java:`, `local:` and `server` namespace dumps

To understand why a naming operation is failing, you can view the dump of a `java:` or `local:` namespace. From the WebSphere Application Server scripting tool, invoke a NameServer MBean to dump `java:` or `local:` namespaces.

Before you begin

Start the naming service.

If the namespaces that you want to view are not local to the server process, use the dumpNameSpace tool.

About this task

You cannot use the dumpNameSpace tool to dump a `java:` or `local:` namespace because the dumpNameSpace tool cannot access those namespaces.

The `java:` namespace of a Java Platform, Enterprise Edition (Java EE) application is accessible only by that application. You can invoke a NameServer MBean to dump the `java:` namespace for any Java EE application running in the same server process.

The `local:` namespace contains references to enterprise beans with local interfaces. There is only one `local:` namespace in a server process. You can invoke the NameServer MBean associated with that server process to dump the `local:` namespace.

Use the scripting tool to invoke the NameServer MBean running in the application's server process to generate dumps of `java:` , `local:`, or `server` namespaces.

1. Invoke a method on a NameServer MBean by using the WebSphere Application Server scripting tool.

   Enter the scripting command prompt by typing the following command:

   ```
   wsadmin
   ```

   Use the `-help` option for help on using the wsadmin command.
2. Select the NameServer MBean instance to invoke.

   Run the following script commands to select the NameServer instance you want to invoke. For example,

```
set mbean [$AdminControl completeObjectName WebSphere:*,type=NameServer,cell=
    cellName,node=nodeName,process=serverName]
```

   where *cellName*, *nodeName*, and *serverName* are the names of the cell, node, and server for the MBean you want to invoke. The specified server must be running before you can invoke a method on the MBean.

   You can see a list of all NameServer MBeans current running by issuing the following query:

   ```
   $AdminControl queryNames {*:*,type=NameServer}
   ```
3. Invoke the NameServer MBean.

   You can dump a `java:` , `local:`, or `server` namespace. For each of these, the value for *opts* is the list of namespace dump options described in "Namespace dump utility for `java:`, `local:` and `server` namespaces" on page 141. The list can be empty.

   java: namespace

   Dump a `java:` namespace by invoking the dumpJavaNameSpace method on the NameServer MBean. Since each server application has its own `java:` namespace, the application must be specified on the method invocation. An application is identified by the application name, module name, and component name. The method syntax follows:

```
$AdminControl invoke $mbean dumpJavaNameSpace {{appname}{modName}{compName}{opts}}
```

where *appname* is the application name, *modName* is the module name, and *compName* is the component name of the `java:` namespace you want to dump.

local: namespace
Dump a `java:` namespace by invoking the dumpLocalNameSpace method on the NameServer MBean. Because there is only one `local:` namespace in a server process, you have to specify the namespace dump options only.

```
$AdminControl invoke $mbean dumpLocalNameSpace {{opts}}
```

Server namespace
Dump a server namespace by invoking the dumpServerNameSpace method on an application server's NameServer MBean. This provides an alternative way to dump the namespace on an application server, much like the dumpNameSpace command line utility.

```
$AdminControl invoke $mbean dumpServerNameSpace {{opts}}
```

Results

Namespace dump output is sent to the console. It is also written to the file DumpNameSpace.log in the server's log directory.

Example

Dumping a `java:` namespace

Assume you want to dump the `java:` namespace of an application component running in server server1 on node node1 of the cell MyCell. The application name is AcctApp in module AcctApp.war, and the component name is Acct Servlet. The following script commands generate a long format dump of the application's `java:` namespace of that application:

```
set mbean [$AdminControl completeObjectName WebSphere:*,type=NameServer,cell=MyCell,node=node1,process=server1]
$AdminControl invoke $mbean dumpJavaNameSpace {{DefaultApplication}{Increment.jar}{Increment}{-report long}}
```

Dumping a `local:` namespace

Assume you want to dump the `local:` namespace for the server server1 on node node1 of cell MyCell. The following script commands generate a short format dump of that server's local namespace:

```
set mbean [$AdminControl completeObjectName WebSphere:*,type=NameServer,cell=MyCell,node=node1,process=server1]
$AdminControl invoke $mbean dumpLocalNameSpace {{-report short}}
```

Using Jython to dump `java:`, `local:` or `server` namespaces

Assume you want to use Jython to run the NameServer MBean methods that dump `java:`, `local:` or `server` namespaces for the server server1 on node node1.

The following script commands set the NameServer instance that you want to invoke to `nameServerString` and then dump a `java:` namespace for DefaultApplication:

```
nameServerString = AdminControl.completeObjectName("WebSphere:type=NameServer,node=node1,process=server1,*")
print AdminControl.invoke(nameServerString, "dumpJavaNameSpace",
          '[DefaultApplication Increment.jar Increment "-report long"]')
```

The following script commands set the NameServer instance that you want to invoke to `nameServerString` and then dump a `local:` namespace:

```
nameServerString = AdminControl.completeObjectName("WebSphere:type=NameServer,node=node1,process=server1,*")
print AdminControl.invoke(nameServerString, "dumpLocalNameSpace", '["-report short"]')
```

The following script commands set the NameServer instance that you want to invoke to `nameServerString` and then dump a `server` namespace:

```
nameServerString = AdminControl.completeObjectName("WebSphere:type=NameServer,node=node1,process=server1,*")
print AdminControl.invoke(nameServerString, "dumpServerNameSpace", '["-root server"]')
```

Namespace dump utility for `java:`, `local:` and `server` namespaces

Sometimes it is helpful to dump the `java:` namespace for a Java Platform, Enterprise Edition (Java EE) application. You cannot use the dumpNameSpace command line utility for this purpose because the application's `java:` namespace is accessible only by that Java EE application. From the product scripting tool, you can invoke a NameServer MBean to dump the `java:` namespace for any Java EE application running in that same server process.

There is another namespace local to server process which you cannot dump with the dumpNameSpace command line utility. This namespace has the URL scheme of local: and is used by the container to bind objects locally instead of through the name server. The `local:` namespace contains references to enterprise beans with local interfaces. There is only one `local:` namespace in a server process. You can dump the `local:` namespace by invoking the NameServer MBean associated with that server process.

Namespace dump options

Namespace dump options are specified in the MBean invocation as a parameter in character string format. The option descriptions follow.

-startAt *some/subcontext/in/the/tree*
Indicates the path from the namespace root context to the top level context where the dump should begin. The utility recursively dumps subcontexts below this point. It defaults to an empty string, that is, the root context.

-report {short | long}

Option	Description
short	The default. Dumps the binding name and bound object type. This output is also provided by Java Naming and Directory Interface (JNDI) Context.list().
long	Dumps the binding name, bound object type, local object type, and string representation of the local object (that is, the IORs, string values, and other values that are printed).

-root {tree | host | legacy | cell | node | server | default}
Specify the root context of where the dump should start. The default value for `-root` is *cell*. This option is only valid for server namespace dumps.

Option	Description
tree	Dump the tree starting at the tree root context.
host	Dump the tree starting at the server host root context (synonymous with ″node″).
legacy	Dump the tree starting at the legacy root context.
cell	Dump the tree starting at the cell root context. This is the default option.
node	Dump the tree starting at the node root context (synonymous with ″host″).
server	Dump the tree starting at the server root context. This is `-root` default.
default	Dump the tree starting at the initial context which JNDI returns by default for that server type.

-format {jndi | ins}

Specify the format to display name component as atomic strings or parsed according to INS rules (id.kind). This option is only valid for server namespace dumps.

Option	Description
jndi	Display name components as atomic strings. This is `-format` default.
ins	Display name components parsed according to INS rules (id.kind).

Chapter 9. Transactions

Troubleshooting transactions

Use this overview task to help resolve a problem that you think is related to the Transaction service.

About this task

To identify and resolve transaction-related problems, you can use the standard WebSphere Application Server RAS facilities. If you encounter a problem that you think might be related to transactions, complete the following steps:

1. Check for transaction messages in the administrative console.

   The Transaction service produces diagnostic messages prefixed by “CWWTR”. The error message indicates the nature of the problem and provides some detail. The associated message information provides an explanation and any user actions to resolve the problem.
2. Check for Transaction messages.

   Check the activity log.
3. Check for more messages in additional places.

   Check the application server’s stdout.log. For more information about a problem, check the stdout.log file for the application server, which should contain more error messages and extra details about the problem.
4. Check for messages related to the application server’s transaction log directory when the problem occurred.

   Note: If you changed the transaction log directory and a problem caused the application server to fail (with in-flight transactions) before the server was restarted properly, the server will next start with the new log directory and be unable to automatically resolve in-flight transactions that were recorded in the old log directory. To resolve this, you can copy the transaction logs to the new directory then stop and restart the application server.
5. Check the hints and tips for troubleshooting transactions.

   See “Transaction troubleshooting tips” for an ongoing collection of troubleshooting tips, based on test and user experience. If you have suggestions for other tips, please let the IBM WebSphere writing team know.

Transaction troubleshooting tips

This topic provides tips to help you troubleshoot problems with the WebSphere Application Server transaction service.

- Peer recovery fails to acquire a lock
- “XAER_NOTA exception logged after server fails” on page 144
- “Clean shutdown message is not in the message log” on page 144

For messaging problems specific to WebSphere Application Server nodes, see other topics in the information center, such as “Messaging troubleshooting tips” on page 76, and the Application Servers support Web site.

Peer recovery fails to acquire a lock

If peer recovery of a transaction fails to acquire a file lock that is needed to perform recovery processing, you should see the following messages:

```
[10/26/04 8:41:38:887 CDT] 00000029 CoordinationL A   CWWTR0100_GENERIC_ERROR
[10/26/04 8:41:39:100 CDT] 00000029 RecoveryHandl A   CWWTR0100E: An attempt to
acquire a file lock need to perform recovery processing failed. Either the target
server is active or the recovery log configuration is incorrect
....
```

```
[10/26/04 8:42:34:921 CDT] 00000027 HAGroupImpl   I   CWRHA0130I: The local member
of group GN_PS=fwsitkaCell01\fwwsaix1Node01\GriffinServer3,IBM_hc=GriffinCluster,type
=WAS_TRANSACTIONS has indicated that is it not alive. The JVM will be terminated.
[10/26/04 8:42:34:927 CDT] 00000027 SystemOut     O Panic:component requested panic
from isAlive
```

To troubleshoot the cause of failure to acquire the file lock, check the following factors:

- If you have enabled failover of transaction log recovery on the server cluster and are using a NAS devise for the transaction logs, check that the DFS™ level on your machine is at a correct level for the NAS DFS level. If the two levels are not correct, the transaction logs cannot be accessed.
- If you are running as non-root, check that the id numbers of the non-root user and group match on all machines involved with peer recovery.
- If you have a policy defined for transaction, review the policy to ensure that you are giving control to the correct servers (perhaps you need to add to or reorder the preferred server list).

XAER_NOTA exception logged after server fails

If an application server fails, and the end transaction record is not forced to disk immediately, you may or may not recover a transaction.

WebSphere Application Server does not force the end record to the log, so it is up to the operating system/network file system to decide when to write to the disk. The record would be forced if the server was shutdown cleanly. The transaction service is designed to cope with the case of the end record never being written to disk - when it gets an XAER_NOTA returned from the databases.

```
[date time] 00000057 WSRdbXaResour E   CWWRA0302E:  XAException occurred.
                                        Error code is: XAER_NOTA (-4).  Exception is: XAER_NOTA
```

If there is a transaction without an end record left in the transaction log, the transaction service tries to check with the database. If the transaction has completed, the database indicates that there is nothing to complete (XAER_NOTA). This is normal behavior, and not an error.

Clean shutdown message is not in the message log

When an application server shuts down, any active transactions are rolled back. If all transactions complete successfully, message CWWTR0105I is logged, indicating a clean shutdown of the transaction service, and the next server restart does not need any recovery activity. If an application server shuts down and message CWWTR0105I is not logged, this does not indicate a problem, but it does mean that recovery activity is required when the server restarts.

Before uninstalling the product, you should have a clean shutdown of all application servers so that you avoid data integrity problems.

Chapter 10. Learn about WebSphere programming extensions

Use this section as a starting point to investigate the WebSphere programming model extensions for enhancing your application development and deployment.

See the *Developing and deploying applications* PDF book for a brief description of each WebSphere extension.

Your applications can use the Eclipse extension framework. Your applications are extensible as soon as you define an extension point and provide the extension processing code for the extensible area of the application. You can also plug an application into another extensible application by defining an extension that adheres to the target extension point requirements. The extension point can find the newly added extension dynamically and the new function is seamlessly integrated in the existing application. It works on a cross Java Platform, Enterprise Edition (Java EE) module basis.

The application extension registry uses the Eclipse plug-in descriptor format and application programming interfaces (APIs) as the standard extensibility mechanism for WebSphere applications. Developers that build WebSphere application modules can use WebSphere Application Server extensions to implement Eclipse tools and to provide plug-in modules to contribute functionality such as actions, tasks, menu items, and links at predefined extension points in the WebSphere application.

ActivitySessions

Troubleshooting ActivitySessions

Use this overview task to help resolve a problem that you think is related to the ActivitySession service.

About this task

To identify and resolve ActivitySession-related problems, you can use the standard WebSphere Application Server RAS facilities. If you encounter a problem that you think might be related to ActivitySessions, complete the following stages:

1. Check for ActivitySession messages in the admin console. The ActivitySession service produces diagnostic messages prefixed by “WACS”. The error message indicates the nature of the problem and provides some detail. The associated message information provides an explanation and any user actions to resolve the problem.
2. Check for ActivitySession messages. The ActivitySession service produces diagnostic messages prefixed by “WACS”. The error message indicates the nature of the problem and provides some detail. The associated message information provides an explanation and any user actions to resolve the problem. Activity log messages produced by the ActivitySession service are accompanied by Log and Trace Analyzer descriptions.

   Check in the application server’s SystemOut log at *was_home*\logs*server*\SystemOut for error messages with the prefix WACS. If needed, check other messages, which should provide extra details about the problem.

Dynamic cache

Troubleshooting the dynamic cache service

This topic includes steps to troubleshoot the dynamic cache service.

About this task

Complete the steps below to resolve problems that you think are related to the dynamic cache service.

1. Review the logs. Messages that are prefaced with *DYNA* result from dynamic cache service operations.
   a. Find any messages prefaced with *DYNA* in the log you are viewing, and write down the message IDs. A sample message having the message ID *DYNA0030E* follows:

```
DYNA0030E: "property" element is missing required attribute "name".
```

   b. Find the message for each message ID in the information center for WebSphere Application Server. In the information center navigation tree, click ***product_name*** **> Reference > Troubleshooter > Messages > DYNA** to view dynamic cache service messages.
   c. Read the message **Explanation** and **User Action** statements. A search for the message ID `DYNA0030E` displays a page having the following message:

```
DYNA0030E: "property" element is missing required attribute "name".
  Explanation: A required attribute was missing in the cache configuration.
  User Action: Add the required attribute to your cache configuration file.
```

   This explanation and user action suggests that you can fix the problem by adding or correcting a required attribute in the cache configuration file.
   d. Try the solutions stated under **User Action** in the *DYNA* messages.
2. Use the cache monitor to determine whether the dynamic cache service is functioning as expected. The cache monitor is an installable Web application that displays simple cache statistics, cache entries, and cache policy information. Read the information about cache monitor in the *Setting up the application serving environment* PDF book.
3. If you have completed the preceding steps and still cannot resolve the problem, contact your IBM software support representative.

   The IBM representative might ask you to complete a diagnostic trace. To enable tracing in the administrative console, click **Troubleshooting > Logs and trace >** ***server_name*** **> Diagnostic trace** and specify **Enable trace with the following specification**. The IBM representative can tell you what trace specification to enter. Note that dynamic cache trace files can become large in a short period of time; you can limit the size of the trace file by starting the trace, immediately recreating the problem, and immediately stopping the trace.

   For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

   For technical support on dynamic cache service, see the IBM Support page.

Troubleshooting tips for the dynamic cache service

The dynamic cache service works within an application server Java virtual machine (JVM), intercepting calls to cacheable objects. This article describes some common runtime and configuration problems and remedies.

Servlets are not cached

Recommended response	Enable servlet caching. On the Web container settings page of the administrative console, select the **Enable servlet caching** check box. Read the information about the Web container settings in the *Developing and deploying applications* PDF book.

Cache entries are not written to disk

Explanation	Cache entries are written to disk when the cache is full and new entries are added to the memory cache. Cache entries also are written to disk when **Flush to disk** is enabled in the administrative console and the server is stopped.

Recommended response

Verify that **Disk offload** is enabled on the dynamic cache service settings page of the administrative console. Also verify that cache entries written to disk are serializable and do not have the `PersistToDisk` configuration set to `false`. Read more information about the dynamic cache settings in the *Developing and deploying applications* PDF book.

Some servlets are not replicated or written to disk

Recommended response

Ensure that the attributes and response are serializable. If you do not want to store the attributes, use the following property in your cache policy:

```
<property name="save-attributes">false</property>
```

Dynamic cache service does not cache fragments on the Edge

Recommended response

Set the `EdgeCacheable` property to `true` in the cache policy for those entries that are to be cached on the Edge.

```
<property name="EdgeCacheable">true</property>
```

Dynamic cache invalidations are not sent to the IBM HTTP Server plug-in

Explanation

The `DynaCacheEsi.ear` file is required to send invalidations to external caches.

Recommended response

Install `DynaCacheEsi.ear` using the administrative console. See the *Developing and deploying applications* PDF for more information.

Cache entries are evicted often

Problem

The cache is full and new entries are added to the cache.

Explanation

Cache entries are evicted when the cache is full and new entries are added to the cache. A least recently used (LRU) eviction mechanism removes the least recently used entry to make space for the new entries.

Recommended response

Enable **Disk offload** on the Dynamic cache service settings page of the administrative console so the entries are written to disk. You can also increase the cache size to accommodate more entries in the cache. Read more information about the dynamic cache settings in the *Developing and deploying applications* PDF book.

Cache entries in disk with timeout set to 0 expire after one day

Explanation

The maximum lifetime of an entry in disk cache is 24 hours. A timeout of `0` in the cache policy configures these entries to stay in disk cache for one whole day, unless they are evicted earlier.

Recommended response

Set the timeout for the cache policy to a number less than `0`.

I cannot monitor cache entries on the Edge

Explanation

Use the dynamic cache monitor to monitor the contents of the memory cache, disk cache and external caches, like the Edge cache. For the ESI processor's cache to be visible in the cache monitor, the `DynaCacheEsi.ear` application must be installed and the `esiInvalidationMonitor` property must be set to `true` in the `plugin-cfg.xml` file.

Recommended response

Set the `esiInvalidationMonitor` property in the `plugin-cfg.xml` file to `true`. See the *Administering applications and their environment* PDF for more information.

I want to tune cache for my environment

Recommended response

Use the Tivoli Performance viewer to study the caching behavior for your applications. Also consider performing the following actions:

- Increase the priority of cache entries that are expensive to regenerate.
- Modify timeout of entries so that they stay in memory as long as they are valid.
- Enable disk offload to store LRU evicted entries.
- Increase the cache size.

Cleaning the disk cache files after installing the fix pack or a new release if you use the disk cache function

Symptom

If the server is configured to use the disk cache, you must delete the disk cache files because the disk cache files are not compatible to the previous version.

Problem

Failure to remove the old disk cache files results in a ClassCastException error in the systemerr.log file when you access the cache from the disk.

Recommended response

To delete the disk cache, perform the following steps:

1. Note your disk offload location. If you do not know the disk cache offload location, perform the following steps:
   a. Click **Servers > Application servers >** *server_name* **> Container services > Dynamic cache service** in the administrative console navigation tree.
   b. The location is specified in the Disk offload field. If the location is not specified, the default directory `*profile_root*/temp/*your_node*/*server_name*/_dynacache` is used.
2. Make sure that the you stop the server and delete all the files under the offload location.
3. If you use the Network Deployment product, delete the disk cache files for each server.

Setting the flush attribute to true on every `<jsp:include>`tag in the cacheable JavaServer Pages file

Symptom

When you obtain the JavaServer Pages (JSP) file from the dynamic cache, a part of the page is not displayed.

Problem

The flush attribute is set to false on the `<jsp: include>` tag in the JSP file.

Description

When the cacheable JSP file includes another JSP file and if the flush attribute is set to false on the `<jsp: include>` tag, any data written to the parent output stream before the`<jsp: include>` tag are not cached.

Recommended response

Set `flush=true` on every `<jsp: include>` tag in the cacheable JSP file.

Dynamic cache limitation when using the JSTL <c:import> tag to include a fragment

Problem

When a cacheable fragment is included using the JavaServer Pages Standard Tag Library (JSTL) `<c:import>` tag, part of the page content disappears and part of the page content displays twice on a cache hit.

Description

Dynamic cache relies on flushing the content before and after including a fragment so that the parent content before the include is not lost, and also to prevent pulling the child content into the parent fragment.

However, in the case of the JSTL `<c:import>` tag, the flush=true attribute, which flushes the parent writer before the child fragment is actually invoked, is not supported. Also, JSTL buffers the responses, so that the child writer is not flushed right after the child fragment is done. Subsequently, the child response is pulled into the parent.

Note: Dynamic cache will return multiple include statements when the JSTL `<c:import>` tag is used in a cached JavaServer Pages (JSP) file.

Recommended response

To avoid this problem, surround the `<c:import>` statement with out.flush method statements as follows:

```
<% out.flush(); %>
<c:import url="DNCParent2.jsp" />
<% out.flush(); %>
```

Appendix. Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories infer specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
: The default installation root directory for the Java EE WebSphere Application Client is the /QIBM/ProdData/WebSphere/AppClient/V7/client directory.

app_client_user_data_root
: The default Java EE WebSphere Application Client user data root is the /QIBM/UserData/WebSphere/AppClient/V7/client directory.

app_client_profile_root
: The default Java EE WebSphere Application Client profile root is the /QIBM/UserData/WebSphere/AppClient/V7/client/profiles/*profile_name* directory.

app_server_root
: The default installation root directory for WebSphere Application Server Network Deployment is the /QIBM/ProdData/WebSphere/AppServer/V7/ND directory.

cip_app_server_root
: The default installation root directory is the /QIBM/ProdData/WebSphere/AppServer/V7/ND/cip/*cip_uid* directory for a customized installation package (CIP) produced by the Installation Factory.

: A CIP is a WebSphere Application Server Network Deployment product bundled with optional maintenance packages, an optional configuration archive, one or more optional enterprise archive files, and other optional files and scripts.

cip_profile_root
: The default profile root directory is the /QIBM/UserData/WebSphere/AppServer/V7/ND/cip/*cip_uid*/profiles/*profile_name* directory for a customized installation package (CIP) produced by the Installation Factory.

cip_user_data_root
: The default user data root directory is the /QIBM/UserData/WebSphere/AppServer/V7/ND/cip/*cip_uid* directory for a customized installation package (CIP) produced by the Installation Factory.

if_root
: This directory represents the root directory of the IBM WebSphere Installation Factory. Because you can download and unpack the Installation Factory to any directory on the file system to which you have write access, this directory's location varies by user. The Installation Factory is an Eclipse-based tool which creates installation packages for installing WebSphere Application Server in a reliable and repeatable way, tailored to your specific needs.

iip_root
: This directory represents the root directory of an *integrated installation package* (IIP) produced by the IBM WebSphere Installation Factory. Because you can create and save an IIP to any directory on the file system to which you have write access, this directory's location varies by user. An IIP is an aggregated installation package created with the Installation Factory that can include one or more generally available installation packages, one or more customized installation packages (CIPs), and other user-specified files and directories.

java_home
: The following directories are the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
Classic JVM	/QIBM/ProdData/Java400/jdk6
32–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root
: The default Web server plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V7/webserver/profiles/*profile_name* directory.

plugins_root
: The default installation root directory for Web server plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V7/webserver directory.

plugins_user_data_root
: The default Web server plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V7/webserver directory.

product_library
product_lib
: This is the product library for the installed product. The product library for each Version 7.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS7*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 7.0 product installed on the system is `QWAS7A`. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, `was.install.library`, and is located under the *app_server_root* directory.

profile_root
: The default directory for a profile named *profile_name* for WebSphere Application Server Network Deployment is the /QIBM/UserData/WebSphere/AppServer/V7/ND/profiles/*profile_name* directory.

shared_product_library
: The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS7`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

updi_root
: The default installation root directory for the Update Installer for WebSphere Software is the /QIBM/ProdData/WebSphere/UpdateInstaller/V7/updi directory.

user_data_root
: The default user data directory for WebSphere Application Server Network Deployment is the /QIBM/UserData/WebSphere/AppServer/V7/ND directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root
: The default web server path is /www/*web_server_name*.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. For a current list of IBM trademarks, visit the IBM Copyright and trademark information Web site (www.ibm.com/legal/copytrade.shtml).

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.