

IBM WebSphere Application Server for z/OS, Version 8.0

*Troubleshooting WebSphere
applications*



Note

Before using this information, be sure to read the general information under “Notices” on page 311.

Compilation date: June 10, 2011

© Copyright IBM Corporation 2011.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

How to send your comments	vii
Changes to serve you more quickly	ix
Chapter 1. Troubleshooting ActivitySessions	1
Troubleshooting ActivitySessions	1
Chapter 2. Troubleshooting Application profiling	3
Application profiling exceptions	3
Chapter 3. Troubleshooting batch applications	5
Troubleshooting batch applications	5
Adding log and trace settings to the batch environment	5
Batch common problems	5
Diagnosing problems using job logs	7
Chapter 4. Troubleshooting applications that use the Bean Validation API	9
Bean validation troubleshooting tips	9
Chapter 5. Troubleshooting Client applications	11
Application client troubleshooting tips	11
Adding tracing and logging for stand-alone clients	16
Chapter 6. Troubleshooting Data access resources	17
Troubleshooting data access problems	17
Data access problems.	17
JDBC trace configuration.	43
Chapter 7. Troubleshooting Dynamic caching	45
Troubleshooting dynamic cache	45
Troubleshooting the dynamic cache service	45
Chapter 8. Troubleshooting EJB applications	53
Troubleshooting Enterprise JavaBeans applications	53
Enterprise bean and EJB container troubleshooting tips	53
Application client log error indicates missing JAR file	53
Enterprise bean cannot be accessed from a servlet, a JSP file, a stand-alone program, or another client	54
Troubleshooting access intents for EJB 2.x entity beans	57
Access intent exceptions.	57
Access intent troubleshooting tips	57
Troubleshooting JPA applications.	59
Logging applications with JPA	62
Troubleshooting JPA deadlocks and transaction timeouts	69
Chapter 9. Troubleshooting Messaging resources	71
Troubleshooting messaging	71
Messaging troubleshooting tips	72
Troubleshooting message-driven beans	78
Troubleshooting performance monitoring statistics	80
Chapter 10. Troubleshooting Naming and directory	81
Troubleshooting namespace problems	81

Naming service troubleshooting tips	81
Application access problems	82
Viewing a namespace dump	86
dumpNameSpace tool.	89
Viewing java:, local:, and server namespace dumps.	91
Namespace dump utility for java:, local: and server namespaces	93
Chapter 11. Troubleshooting Object Request Broker (ORB).	95
Troubleshooting Object Request Brokers	95
Object request broker component troubleshooting tips	95
Chapter 13. Troubleshooting security	99
Troubleshooting security configurations	99
Security components troubleshooting tips.	99
Security configuration and enablement errors	112
Security enablement followed by errors	115
Access problems after enabling security.	122
SSL errors for security	125
Single sign-on configuration troubleshooting tips for security	129
Security authorization provider troubleshooting tips.	132
SPNEGO trust association interceptor (TAI) troubleshooting tips (deprecated)	136
SPNEGO troubleshooting tips	143
Chapter 14. Troubleshooting Service integration	153
Troubleshooting service integration technologies	153
Resolving indoubt transactions	155
Restoring a data store and recovering its messaging engine	156
Problem solving for messaging engine file stores	158
Diagnosing problems with accessing file store files.	158
Reducing file store file sizes	159
Problem solving for messaging engine data stores	160
Diagnosing problems with data store exclusive access locks	160
Fixing storage allocation error caused by DB2 Universal JDBC Driver on z/OS	161
Diagnosing problems with your data store configuration	162
Avoiding failover problems when you use DB2 v8.2 with HADR as your data store	162
Listing messages on a message point	163
Deleting messages on a message point.	163
Troubleshooting service integration message problems	164
Understanding why best effort nonpersistent messages are being discarded	164
Investigating why a queue is full	165
Investigating why a topic space is full	166
Investigating why point-to-point messages are not arriving	168
Investigating why point-to-point messages are not being consumed	173
Investigating why publish/subscribe messages are not arriving at a subscription	181
Chapter 22. Troubleshooting Session Initiation Protocol (SIP) applications	211
Troubleshooting SIP applications	211
Tracing a SIP container.	212
Chapter 23. Troubleshooting Transactions	215
Troubleshooting transactions	215
Transaction troubleshooting tips.	216
Transaction service exceptions	217
Transaction exceptions that involve both one-phase and two-phase commit resources.	218
Chapter 24. Troubleshooting web applications	219

Troubleshooting web applications	219
Web application deployment troubleshooting tips	219
JavaServer Pages troubleshooting tips	220
Troubleshooting contexts and dependency injection	223
Troubleshooting HTTP sessions.	226
HTTP session manager troubleshooting tips	226
HTTP session problems	227
Chapter 25. Troubleshooting web services.	231
Troubleshooting web services	231
Directory conventions	231
Web services command-line tools troubleshooting tips	232
Web services compiled bindings troubleshooting tips	236
Web services client runtime troubleshooting tips.	237
Web services serialization and deserialization troubleshooting tips	240
Web services authentication, authorization and secure transport troubleshooting tips	241
Application client SOAP request troubleshooting tips	243
Universal Discovery, Description, and Integration, web service, and SOAP component troubleshooting tips	244
Tracing web services.	244
Tracing SOAP messages with tcpmon	246
Frequently asked questions about web services.	247
Web Services Security troubleshooting tips	249
Detecting and fixing problems with WS-ReliableMessaging	263
WS-ReliableMessaging sequence reallocation	265
Diagnosing the problem when a reliable messaging source cannot deliver its messages	266
Diagnosing and recovering a WS-ReliableMessaging outbound sequence that is in retransmitting state	266
Deleting a failed WS-ReliableMessaging outbound sequence	267
WS-ReliableMessaging troubleshooting tips	268
Troubleshooting WSIF	273
Tracing and logging WSIF	274
WSIF (Web Services Invocation Framework) messages	275
Web Services Invocation Framework troubleshooting tips	276
WSIF - Known restrictions.	280
UDDI registry troubleshooting	281
Bus-enabled web services troubleshooting tips	283
Bus-enabled web services: Known restrictions	289
Web services gateway troubleshooting tips	291
WS-Notification troubleshooting tips	296
WS-Notification: Known restrictions	306
Appendix. Directory conventions	309
Notices	311
Trademarks and service marks	313
Index	315

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. Troubleshooting ActivitySessions

This page provides a starting point for finding information about ActivitySessions, a WebSphere extension for reducing the complexity of commitment rules and limitations that are associated with one-phase commit resources.

Use ActivitySessions to extend the scope and group multiple local transactions. With this capability, you can commit these transactions based on either deployment criteria or through explicit program logic. More introduction...

Troubleshooting ActivitySessions

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

About this task

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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:

Procedure

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 SYSPRINT or SYSOUT log for error messages with the prefix WACS. If needed, check other messages, which should provide extra details about the problem.

Chapter 2. Troubleshooting Application profiling

This page provides a starting point for finding information about application profiling, a WebSphere extension for defining strategies to dynamically control concurrency, prefetch, and read-ahead.

Application profiling and access intent provide a flexible method to fine-tune application performance for enterprise beans without impacting source code. Different enterprise beans, and even different methods in one enterprise bean, can have their own intent to access resources. Profiling the components based on their access intent increases performance in the application server run time.

Application profiling exceptions

The following exceptions are thrown in response to various illegal actions related to application profiling.

com.ibm.ws.exception.RuntimeWarning

This exception is thrown when the application is started, if the application is configured incorrectly. The startup is consequently terminated. Some examples of bad configurations include:

- A task configured on two different application profiles.
- A method configured with two different task run-as policies .

com.ibm.websphere.appprofile.IllegalTaskNameException

This exception is raised if an application attempts to programmatically set a task when that task has not been configured as a task name reference.

Chapter 3. Troubleshooting batch applications

You can troubleshoot batch application issues using logging and tracing, or reviewing solutions to problems.

Troubleshooting batch applications

You can troubleshoot batch application issues using such things as messages and logging and tracing.

Adding log and trace settings to the batch environment

The batch environment uses the WebSphere Application Server logging and tracing system.

Log and trace settings

Specify the following settings depending on the component:

Table 1. Settings for logging and tracing. The table includes the component in the description column and the settings for the component.

Description	Setting
Scheduler	com.ibm.ws.batch.*=all com.ibm.ws.grid.* = all com.ibm.ws.gridcontainer.*=all
Endpoints	com.ibm.ws.batch.*=all com.ibm.ws.ci.* = all com.ibm.ws.grid.* = all com.ibm.ws.gridcontainer.*=all

Location of log and trace files

Table 2. Log and trace files. The table includes the component in the description column and the location of the log and trace files for the component in the location column.

Description	Location
Schedulers	<user_install_root>/logs/<server_name>
Endpoints	<user-install-root>/logs/<server_name>

Batch common problems

Occasionally, you might encounter behavior in the batch component that is not expected.

Troubleshooting

Use this section to look for solutions to problems when batch is not working, or not working the way that you expect it to.

Job submission fails due to database failures with the default Apache Derby database

- Check for the successful creation of the LRSCHED database in the <user_install_root>/gridDatabase directory.
- Check the file permissions of the database.

- Derby is only supported on a single scheduler configuration. Use a shared RDBMS for cells configured with more than one scheduler. For example, DB2®.

Job submission fails with the following message: Unable to submit the job definition <xJCL file> because the application that it runs has not been deployed to an endpoint

- Ensure that the application is installed on an endpoint server.
- Ensure the job name or the application name specified in the XJCL matches the name of the application.

Job dispatching slowly when large number of jobs (hundreds or thousands) are submitted

Increase the number of dispatcher threads by setting the MaxConcurrentDispatchers custom property in the job scheduler custom properties panel in the administrative console.

Job execution fails due to database failures with the default Derby database

- Check for the successful creation of the LRSCHED database in the <user_install_root>/gridDatabase directory
- Check the file permissions of the database.

Database errors during the execution of batch jobs with DB2

- Check for the successful creation of the LRSCHED database.
- Do not use default the Derby data source JNDI name (jdbc/lree) with DB2. Create a data source for non-default Derby databases.
- Check that the WebSphere variable of GRID_ENDPOINT_DATASOURCE is set to the newly created non-default data source.

Jobs are creating files with the server identity

Set the WebSphere variable RUN_JOBS_UNDER_USER_CREDENTIAL to run jobs under the credential of the submitter. Although jobs can run under the credential of the user on distributed and z/OS® operating systems, they work slightly differently. On distributed operating systems, files are created with the identity of the server even if the thread has the credential of the user. On z/OS, the Java thread synchronizes with the operating system thread and files are created with the identity of the user.

Batch applications not working with Java 2 Security

Set the WebSphere variable RUN_JOBS_UNDER_USER_CREDENTIAL to run jobs under the credential of the submitter. Although jobs can run under the credential of the user on distributed and z/OS operating systems, they work slightly differently. On distributed operating systems, files are created with the identity of the server even if the thread has the credential of the user. On z/OS, the Java thread synchronizes with the operating system thread and files are created with the identity of the user.

- Ensure that application security is turned on.
- Grant permissions, SecOwnCredentials, and ContextManager.getServerCredential, in the policy file of the application.

Job log viewing from the Job Management Console fails with the following error: Unable to read the job log.

If administrative security is enabled, ensure that application security is also enabled.

Diagnosing problems using job logs

Occasionally, you might encounter behavior in the batch component that is not expected.

CWLRB5586I

Job remains in submitted state with the following message: CWLRB5586I: Job cannot be dispatched at this time. Waiting for an endpoint, a job application, or both to become active.

- Ensure that the application and endpoint servers are running.

CWLRB3112E

Job remains in submitted state with the following message: CWLRB3112E: Job could not be dispatched. Required capability was not found.

- Ensure the required-capability operands are configured correctly.
- If the problem remains, recycle the scheduler and endpoint servers.

Chapter 4. Troubleshooting applications that use the Bean Validation API

The Bean Validation API is introduced with the Java Enterprise Edition 6 platform as a standard mechanism to validate Enterprise JavaBeans in all layers of an application, including, presentation, business and data access. Before the Bean Validation specification, the JavaBeans were validated in each layer. To prevent the reimplementations of validations at each layer, developers bundled validations directly into their classes or copied validation code, which was often cluttered. Having one implementation that is common to all layers of the application simplifies the developers work and saves time.

Bean validation troubleshooting tips

Use this information to troubleshoot bean validation.

An application module does not start when a validation factory cannot be created.

There are various problems related to bean validation and the validation.xml file that might prevent the validation factory from being created and the application module to not start. The following messages that are logged in this case:

- An incorrect syntax or error is detected in the validation.xml file found in *module_name*. The following associated error message occurred: *error_message*

The message is caused by errors in the validation.xml file most likely because it does not conform to the version 1.0 validation schema definition.

- The BeanValidationService service is unable to create a ValidationFactory class because it was unable to load or instantiate the class, *class_name* in path, *path_name*, because the following error: *error_message*

This error occurs if any of the configuration setting classes that are defined in the validation.xml file are missing or unable to be loaded. Ensure that the class is available and on the application class path. Other class loader issues can also cause this error to occur. For more information about class loading, see the Class loader documentation.

- The BeanValidationService service is unable to create a ValidatorFactory class.

This error might occur if any of the constraint mapping files that are defined in the validation.xml file are not found in the module. Ensure that the defined constraint mapping XML files are available.

Chapter 5. Troubleshooting Client applications

This page provides a starting point for finding information about application clients and client applications. Application clients provide a framework on which application code runs, so that your client applications can access information on the application server.

For example, an insurance company can use application clients to help offload work on the server and to perform specific tasks. Suppose an insurance agent wants to access and compile daily reports. The reports are based on insurance rates that are located on the server. The agent can use application clients to access the application server where the insurance rates are located. More introduction...

Application client troubleshooting tips

This topic provides debugging tips for resolving common Java Platform, Enterprise Edition (Java EE) application client problems. To use this troubleshooting guide, review the trace entries for one of the Java EE application client exceptions, and then locate the exception in the guide.

Some of the errors in the guide are samples, and the actual error you receive can be different than what is shown here. You might find it useful to rerun the `launchClient` command specifying the `-CCverbose=true` option. This option provides additional information when the Java EE application client run time is initializing.

Error: `java.lang.NoClassDefFoundError`

Explanation	This exception is thrown when Java code cannot load the specified class.
Possible causes	<ul style="list-style-type: none">• Invalid or non-existent class• Class path problem• Manifest problem

Recommended response

Check to determine if the specified class exists in a Java Archive (JAR) file within your Enterprise Archive (EAR) file. If it does, make sure the path for the class is correct. For example, if you get the exception:

```
java.lang.NoClassDefFoundError:  
WebSphereSamples.HelloEJB.HelloHome
```

verify that the HelloHome class exists in one of the JAR files in your EAR file. If it exists, verify that the path for the class is WebSphereSamples.HelloEJB.

If both the class and path are correct, then it is a class path issue. Most likely, you do not have the failing class JAR file specified in the client JAR file manifest. To verify this situation, perform the following steps:

1. Open your EAR file with an assembly tool, and select the Application Client.
2. Add the names of the other JAR files in the EAR file to the Classpath field.

This exception is generally caused by a missing Enterprise Java Beans (EJB) module name from the Classpath field.

If you have multiple JAR files to enter in the Classpath field, be sure to separate the JAR names with spaces.

If you still have the problem, you have a situation where a class is loaded from the file system instead of the EAR file. This error is difficult to debug because the offending class is not the one specified in the exception. Instead, another class is loaded from the file system before the one specified in the exception. To correct this error, review the class paths specified with the -CCclasspath option and the class paths configured with the Application Client Resource Configuration Tool.

Alternatively, you can use the Client Container Resource Configuration Scripting tool.

Look for classes that also exist in the EAR file. You must resolve the situation where one of the classes is found on the file system instead of in the EAR file. Remove entries from the classpaths, or include the JAR files and classes in the EAR file instead of referencing them from the file system.

If you use the -CCclasspath parameter or resource classpaths in the tool, and you have configured multiple JAR files or classes, verify they are separated with the correct character for your operating system. Unlike the Classpath field, these class path fields use platform-specific separator characters.

Tip: The system class path is not used by the Application Client run time if you use the launchClient batch or shell files. In this case, the system class path would not cause this problem. However, if you load the launchClient class directly, you do have to search through the system class path as well.

Error: com.ibm.websphere.naming.CannotInstantiateObjectException: Exception occurred while attempting to get an instance of the object for the specified reference object. [Root exception is javax.naming.NameNotFoundException: xxxxxxxxxx]

Explanation

This exception occurs when you perform a lookup on an object that is not installed on the host server. Your program can look up the name in the local client Java Naming and Directory Interface (JNDI) name space, but received a NameNotFoundException exception because it is not located on the host server. One typical example is looking up an EJB component that is not installed on the host server that you access. This exception might also occur if the JNDI name you configured in your Application Client module does not match the actual JNDI name of the resource on the host server.

Possible causes

- Incorrect host server invoked
- Resource is not defined
- Resource is not installed
- Application server is not started
- Invalid JNDI configuration

Recommended response

If you are accessing the wrong host server, run the `launchClient` command again with the `-CCBootstrapHost` parameter specifying the correct host server name. If you are accessing the correct host server, use the product `dumpnamespace` command line tool to see a listing of the host server JNDI name space. If you do not see the failing object name, the resource is either not installed on the host server or the appropriate application server is not started. If you determine the resource is already installed and started, your JNDI name in your client application does not match the global JNDI name on the host server. Use the Application Server Toolkit to compare the JNDI bindings value of the failing object name in the client application to the JNDI bindings value of the object in the host server application. The values must match.

Error: javax.naming.ServiceUnavailableException: A communication failure occurred while attempting to obtain an initial context using the provider url: "iiop://[invalidhostname]". 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.INTERNAL: JORB0050E: In Profile.getIPAddress(), InetAddress.getByName[invalidhostname] threw an UnknownHostException. minor code: 4942F5B6 completed: Maybe

Explanation

This exception occurs when you specify an invalid host server name.

Possible causes

- Incorrect host server invoked
- Invalid host server name

Recommended response

Run the `launchClient` command again and specify the correct name of your host server with the `-CCBootstrapHost` parameter.

Error: javax.naming.CommunicationException: Could not obtain an initial context due to a communication failure. Since no provider URL was specified, either the bootstrap host and port of an existing ORB was used, or a new ORB instance was created and initialized with the default bootstrap host of "localhost" and the default bootstrap port of 2809. Make sure the ORB bootstrap host and port resolve to a running name server. Root exception is org.omg.CORBA.COMM_FAILURE: WRITE_ERROR_SEND_1 minor code: 49421050 completed: No

Explanation

This exception occurs when you run the `launchClient` command to a host server that does not have the Application Server started. You also receive this exception when you specify an invalid host server name. This situation might occur if you do not specify a host server name when you run the `launchClient` tool. The default behavior is for the `launchClient` tool to run to the local host, because WebSphere Application Server does not know the name of your host server. This default behavior only works when you are running the client on the same machine with WebSphere Application Server is installed.

Possible causes

- Incorrect host server invoked
- Invalid host server name
- Invalid reference to `localhost`
- Application server is not started
- Invalid bootstrap port

Recommended response

If you are not running to the correct host server, run the `launchClient` command again and specify the name of your host server with the `-CCBootstrapHost` parameter. Otherwise, start the Application Server on the host server and run the `launchClient` command again.

Error: javax.naming.NameNotFoundException: Name comp/env/ejb not found in context "java:"

Explanation

This exception is thrown when the Java code cannot locate the specified name in the local JNDI name space.

Possible causes

- No binding information for the specified name
- Binding information for the specified name is incorrect
- Wrong class loader was used to load one of the program classes
- A resource reference does not include any client configuration information
- A client container on the deployment manager is trying to use enterprise extensions (not supported)

Recommended response

Open the EAR file with the Application Server Toolkit, and check the bindings for the failing name. Ensure this information is correct. If you are using Resource References, open the EAR file with the Application Client Resource Configuration Tool, and verify that the Resource Reference has client configuration information and the name of the Resource Reference exactly matches the JNDI name of the client configuration. If the values are correct, you might have a class loader error. For detailed information about the configuration tool and opening EAR files, read the Starting the Application Client Resource Configuration Tool in the *Developing and deploying applications* PDF book.

**Error: java.lang.ClassCastException: Unable to load class:
org.omg.stub.WebSphereSamples.HelloEJB._HelloHome_Stub at
com.ibm.rmi.javax.rmi.PortableRemoteObject.narrow(portableRemoteObject.java:269)**

Explanation	This exception occurs when the application program attempts to narrow to the EJB home class and the class loaders cannot find the EJB client side bindings.
Possible causes	<ul style="list-style-type: none">• The files, *_Stub.class and _Tie.class, are not in the EJB .jar file• Class loader could not find the classes
Recommended response	Look at the EJB JAR file located in the EAR file and verify the class contains the Enterprise Java Beans (EJB) client side bindings. These are class files with file names that end in _Stub and _Tie. If the binding classes are in the EJB JAR file, then you might have a class loader error.

**Error: WSCL0210E: The Enterprise archive file [EAR file name] could not be found.
com.ibm.websphere.client.applicationclient.ClientContainerException:
com.ibm.etools.archive.exception.OpenFailureException**

Explanation	This error occurs when the application client run time cannot read the Enterprise Archive (EAR) file.
Possible causes	The most likely cause of this error is that the system cannot find the EAR file in the path specified on the launchClient command.
Recommended response	Verify that the path and file name specified on the launchclient command are correct.

The launchClient command appears to hang and does not return to the command line when the client application has finished.

Explanation	When running your application client using the launchClient command the WebSphere Application Server run time might need to display the security login dialog. To display this dialog, WebSphere Application Server run time creates an Abstract Window Toolkit (AWT) thread. When your application returns from its main method to the application client run time, the application client run time attempts to return to the operating system and end the Java virtual machine (JVM) code. However, since there is an AWT thread, the JVM code will not end until System.exit is called.
Possible causes	The JVM code does not end because there is an AWT thread. Java code requires that System.exit() be called to end AWT threads.
Recommended response	<ul style="list-style-type: none">• Modify your application to call System.exit(0) as the last statement.• Use the -CCexitVM=true parameter when you call the launchClient command.

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/software/webservers/appserv/zos_os390/support/

Adding tracing and logging for stand-alone clients

You can add tracing and logging to help analyze performance and diagnose problems with stand-alone clients.

About this task

This information applies to the following WebSphere Application Server stand-alone clients:

- Thin Client for JMS with WebSphere Application Server
- Thin Client for EJB with WebSphere Application Server
- Thin Client for JAX-WS with WebSphere Application Server
- Thin Client for JAX-RPC with WebSphere Application Server

Procedure

- To enable trace, use either a long form or short form system property.

Note: Trace settings are determined from the system property values the first time that a WebSphere Application Server client is called. The trace settings are then fixed. Therefore, any subsequent changes to the system property settings do not change the trace settings that the WebSphere Application Server client uses.

For further information see `../ae/cjj_stdalonecli_trace.dita`.

- To enable First Failure Data Capture (FFDC), use either a long or short form system property.

Note: FFDC settings are determined from the system property values the first time that a WebSphere Application Server client performs an FFDC. The FFDC settings are then fixed. Therefore, any subsequent changes to the system property settings do not change the FFDC settings that the WebSphere Application Server client uses.

For further information see `../ae/cjj_stdalonecli_log.dita`.

Chapter 6. Troubleshooting Data access resources

This page provides a starting point for finding information about data access. Various enterprise information systems (EIS) use different methods for storing data. These backend data stores might be relational databases, procedural transaction programs, or object-oriented databases.

The flexible IBM WebSphere Application Server provides several options for accessing an information system backend data store:

- Programming directly to the database through the JDBC 4.0 API, JDBC 3.0 API, or JDBC 2.0 optional package API.
- Programming to the procedural backend transaction through various J2EE Connector Architecture (JCA) 1.0 or 1.5 compliant connectors.
- Programming in the bean-managed persistence (BMP) bean or servlets indirectly accessing the backend store through either the JDBC API or JCA-compliant connectors.
- Using container-managed persistence (CMP) beans.
- Using the IBM data access beans, which also use the JDBC API, but give you a rich set of features and function that hide much of the complexity associated with accessing relational databases.

Service Data Objects (SDO) simplify the programmer experience with a universal abstraction for messages and data, whether the programmer thinks of data in terms of XML documents or Java objects. For programmers, SDOs eliminate the complexity of the underlying data access technology such as, JDBC, RMI/IIOP, JAX-RPC, and JMS, and message transport technology such as, `java.io.Serializable`, DOM Objects, SOAP, and JMS.

Troubleshooting data access problems

Data access problems

WebSphere Application Server diagnostic tools provide services to help troubleshoot database connection problems. Additionally, the IBM website provides flexible searching capabilities for finding documented solutions to database-specific connection problems.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

The following steps help you quickly isolate connectivity problems.

1. Browse the log files of the application server for clues.
See the topic, [Setting up the error log](#).
2. Browse the Helper Class property of the data source to verify that it is correct and that it is on the WebSphere Application Server class path. Mysterious errors or behavior might result from a missing or misnamed Helper Class name. If WebSphere Application Server cannot load the specified class, it uses a default helper class that might not function correctly with your database manager.
3. Verify that the Java Naming and Directory Interface (JNDI) name of the data source matches the name used by the client attempting to access it. If error messages indicate that the problem might be naming-related, such as referring to the **name server** or **naming service**, or including error IDs beginning with **NMSV**, look at the topics, [Naming related problems](#), and [Troubleshooting the naming service component](#).

4. Enable tracing for the resource adapter using the trace specification, `RRA=all=enabled`. Follow the instructions for dumping and browsing the trace output, to narrow the origin of the problem. See the topic, [Enabling tracing](#).

For a comprehensive list of database-specific troubleshooting tips, see the [WebSphere Application Server product support page](#). (Find the link at the end of this article.) In the Search Support field, type a database vendor name among your search terms. Select **Solve a problem**, then click **Search**.

Remember that you can always find Support references in the topic, [Troubleshooting help from IBM](#), in this information center.

Currently this information center provides a limited number of troubleshooting tips for the following databases:

- Oracle
- DB2
- SQL Server
- Apache Derby
- Sybase
- General data access problems

General data access problems

- An exception "IllegalConnectionUseException" occurs
- WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred.
- ConnectionWaitTimeoutException.
- `com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name "NULL" could not be found. SQLSTATE=42705`
- `java.sql.SQLException: java.lang.UnsatisfiedLinkError:`
- "J2CA0030E: Method enlist caught java.lang.IllegalStateException" wrapped in error "WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred" when attempting to execute a transaction.
- `java.lang.UnsatisfiedLinkError:xaConnect` exception when attempting a database operation
- "J2CA0114W: No container-managed authentication alias found for connection factory or data source *datasource*" when attempting a database operation
- An error is thrown if you use the `ws_ant` command to perform the database customization for Structured Query Language in Java on HP platforms
- Container-managed persistence (CMP) cannot successfully obtain the database access function as defined.

IllegalConnectionUseException

This error can occur because a connection obtained from a `WAS40DataSource` is being used on more than one thread. This usage violates the J2EE 1.3 programming model, and an exception generates when it is detected on the server. This problem occurs for users accessing a data source through servlets or bean-managed persistence (BMP) enterprise beans.

To confirm this problem, examine the code for connection sharing. Code can inadvertently cause sharing by not following the programming model recommendations, for example by storing a connection in an instance variable in a servlet, which can cause use of the connection on multiple threads at the same time.

WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred

This error can occur because:

- An attempt was made to share a single-phase connection, when each `getConnection` method has different connection properties; such as the `AccessIntent`. This attempt causes a non-shareable connection to be created.

- An attempt was made to have more than one unshareable connection participate in a global transaction, when the data source is not an XA resource.
- An attempt was made to have a one-phase resource participate in a global transaction while an XA resource or another one-phase resource already participated in this global transaction.
 - Within the scope of a global transaction, you try to get a connection more than one time and at least one of the resource-refs you use specifies that the connection is unshareable, and the data source is not configured to support two-phase commit transactions. It does not support an XAResource. If you do not use a resource-ref, you default to unshareable connections.
 - Within the scope of a global transaction, you try to get a connection more than one time and at least one of the resource-refs you use specifies that the connection is shareable and the data source is not configured to support two-phase commit transactions. That is, it does not support an XAResource. In addition, even though you specify that connections are shareable, each getConnection request is made with different connection properties (such as IsolationLevel or AccessIntent). In this case, the connections are not shareable, and multiple connections are handed back.
 - Multiple components (servlets, session beans, BMP entity beans, or CMP entity beans) are accessed within a global transaction. All use the same data source, all specify shareable connections on their resource-refs, and you expect them to all share the same connection. If the properties are different, you get multiple connections. AccessIntent settings on CMP beans change their properties. To share a connection, the AccessIntent setting must be the same. For more information about CMP beans sharing a connection with non-CMP components, see the *Data access application programming interface support* and *Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans* topics in the DataAccess section of the information center.

To correct this error:

- Check what your client code passes in with its getConnection requests, to ensure that they are consistent with each other.
- Check the connection sharing scope from the resource binding, using an assembly tool. See the topic, *Assembly tools*.
 - If you are running an unshareable connection scope, verify that your data source is an XA data source.
 - If you are running a shareable connection scope, verify that all connection properties, including AccessIntent, are shareable.
- Check the JDBC provider implementation class from the Manage JDBC resource panel of the administrative console to ensure that it is a class that supports XA-type transactions.

ConnectionWaitTimeoutException accessing a data source or resource adapter

If your application receives exceptions like a `com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException` or `com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException` when attempting to access a WebSphere Application Server data source or JCA-compliant resource adapter, some possible causes are:

- The maximum number of connections for a given pool is set too low. The demand for concurrent use of connections is greater than the configured maximum value for the connection pool. One indication that this situation is the problem is that you receive these exceptions regularly, but your CPU utilization is not high. This exception indicates that there are too few connections available to keep the threads in the server busy.
- Connection Wait Time is set too low. Current[®] demand for connections is high enough such that sometimes there is not an available connection for short periods of time. If your connection wait timeout value is too low, you might timeout shortly before a user returns a connection back to the pool. Adjusting the connection wait time can give you some relief. One indication of this problem is that you use close to the maximum number of connections for an extended period and receiving this error regularly.
- You are not closing some connections or you are returning connections back to the pool at a slow rate. This situation can happen when using unshareable connections, when you forget to close them, or you close them long after you are finished with them, keeping the connection from returning to the pool for

reuse. The pool soon becomes empty and all applications get `ConnectionWaitTimeoutExceptions`. One indication of this problem is you run out of connections in the connection pool and you receive this error on most requests.

- You are driving more load than the server or backend system have resources to handle. In this case you must determine which resources you need more of and upgrade configurations or hardware to address the need. One indication of this problem is that the application server or the database server processor is nearly 100% busy.

To correct these problems, either:

- Modify an application to use fewer connections
- Properly close the connections.
- Change the pool settings of `MaxConnections` or `ConnectionWaitTimeout`.
- Adjust resources and their configurations.

com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name "NULL" could not be found. SQLSTATE=42705

This error occurs when a data source is defined but the `databaseName` attribute and the corresponding value are not added to the custom properties panel.

To add the `databaseName` property:

1. Click **Resources > Manage JDBC Providers** in the administrative console.
2. Select the `JDBC_provider` that supports the problem data source.
3. Select **Data Sources** and then select the problem data source.
4. Under **Additional properties**, click **Custom Properties**.
5. Select the `databaseName` property, or add one if it does not exist, and enter the actual database name as the value.
6. Click **Apply** or **OK**, and then click **Save** from the action bar.
7. Access the data source again.

java.sql.SQLException: java.lang.UnsatisfiedLinkError:

This error indicates that the directory containing the binary libraries which support a database are not included in the `LIBPATH` environment variable for the environment in which the WebSphere Application Server starts.

The path containing the DBM vendor libraries vary by dbm. One way to find them is by scanning for the missing library specified in the error message. Then you can correct the `LIBPATH` variable to include the missing directory, either in the `.profile` of the account from which WebSphere Application Server is started, or by adding a statement in a `.sh` file which then starts the `startServer` program.

Configure the Java `LIBPATH` (`java.library.path`) property with the `domain_region_libpath` environment variable, like `control_region_libpath`, `server_region_libpath`, `adjunct_region_libpath`. See the topic on changing the values of variables referenced in BBOM00011 messages for instructions on how to set the region libpath variables.

"J2CA0030E: Method enlist caught java.lang.IllegalStateException" wrapped in error "WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred" when attempting to execute a transaction.

This error can occur when last participant support is missing or disabled. Last participant support supports a one-phase capable resource and a two-phase capable resource to enlist within the same transaction.

Last participant support is only available if the following are true:

- WebSphere Application Server Programming Model Extensions (PME) is installed. PME is included in the Application Server Integration Server product.
- The Additional Integration Server Extensions option is enabled when PME is installed. If you perform a typical installation, this function is enabled by default. If you perform a custom installation, you can disable this function, which disables last participant support.
- The application enlisting the one-phase resource is deployed with the **Accept heuristic hazard** option enabled. This deployment is done with an assembly tool.

This problem has two main causes:

- The most common cause is that the JDBC driver that supports connectivity to the database is missing, or is not the correct version. Another common cause is the native libraries which support the driver are on the system path.
 - To resolve this problem on a Windows platform, verify that the JDBC driver JAR file is on the system PATH environment variable:
 - If you are using DB2, verify that at least the DB2 client product has been installed on the WebSphere host
 - On DB2 version 7.2 or earlier, the file where the client product is installed on the WebSphere Application Server is db2java.zip. Verify that the usejdbc2.bat program has been executed after the database install and after any upgrade to the database product.
 - On DB2 version 8.1 or later, use the DB2 Universal JDBC Provider Driver when defining a JDBC provider in WebSphere Application Server. The driver file is db2jcc.jar. If you use the type 2 (default) option, verify that at least the DB2 client product is installed on the WebSphere Application Server host. If you specify the type 4 option, the DB2 client does not need to be installed, but the file db2jcc.jar still must be present.
 - When specifying the location of the driver file, it is recommended that you specify the path and file name of the target DB2 installation, rather than copying the file to a local directory, if possible. Otherwise, you might be exposed to problems if the target DB2 installation is upgraded and the driver used by WebSphere Application Server is not.
 - On operating systems such as AIX® or Linux, ensure that any native libraries required to support the database client of your database product are specified in the LD_LIBRARY_PATH environment variable in the profile of the account under which WebSphere Application Server starts.
 - If you are using DB2 The native library is libdb2jdbc.so. The best way to ensure that this library is accessed correctly by WebSphere is to call the db2profile script supplied with DB2 from the .profile script of the account (such as "root") under which WebSphere runs.
 - If you are using DB2 version 7.2 or earlier, ensure that the usejdbc2,script provided with DB2 is called from the profile of the account under which WebSphere Application Server is launched.
 - If you are using DB2 version 8.1 or later, see the previous instructions for the Windows operating system.
- If the database manager is DB2, you can choose the option to create a 64-bit instance. Sometimes a 64-bit configuration is not supported. If this has happened, remove the database instance and create one with the default 32-bit setting.

If you are using the Universal JDBC T2 driver, WebSphere Application Server does support interaction with a DB2 UDB 64-bit server, but it must be through a DB2 UDB 32-bit client. The WebSphere Application Server environment (CLASSPATH, and so on) must use the 32-bit client code to ensure correct function.

With a Universal JDBC T4 driver, you do not need the 32-bit DB2 client. You need only configure the class path to include db2jcc.jar and its license files in the WebSphere Application Server environment.

Note: For general help in configuring JDBC drivers and data sources in WebSphere Application Server, see the topic, Accessing data from applications.

"J2CA0114W: No container-managed authentication alias found for connection factory or data source *datasource*" when attempting a database operation

This error might occur in the SystemOut.log file when you run an application to access a data source after creating the data source using JACL script.

The error message occurs because the JACL script did not set container-managed authentication alias for CMP connection factory. The JACL is missing the following line:

```
$AdminConfig create MappingModule $cmpConnectorFactory "{mappingConfigAlias  
DefaultPrincipalMapping} {authDataAlias $authDataAlias}
```

To correct this problem, add the missing line to the JACL script and run the script again. See the topic, Example: Creating a JDBC provider and data source using Java Management Extensions API and the scripting tool, for a sample JACL script.

An error is thrown if you use the `ws_ant` command to perform the database customization for Structured Query Language in Java on HP platforms

If you use the `ws_ant` command to perform the database customization for Structured Query Language in Java (SQLJ) on HP platforms, you can receive an error similar to the following:

```
[java] [ibm][db2][jcc][sqlj]  
[java] [ibm][db2][jcc][sqlj] Begin customization  
[java] [ibm][db2][jcc][sqlj] encoding not supported!!
```

The cause of this error might be that your databases were created using the HP default character set. The Java Common Client (JCC) driver depends on the software development kit (SDK) to perform the code page conversions. The SDK shipped with this product, however, does not support the HP default code page.

You must set your LANG to the ISO locale before creating the databases. It should be similar to the following:

```
export LANG=en_US.iso88591
```

Refer to the IBM support site for Information Management software to access the latest technotes for DB2.

Container-managed persistence (CMP) cannot successfully obtain the database access function as defined.

When WebSphere Application Server is caching certain generated code that is accessed in the database on the connection factory, and if any changes in the Java archive (JAR) file require regeneration of the database access, the changes are not effective until you stop and restart the server.

Examples of when this failure might occur include:

- Adding an enterprise bean custom finder method; a `NullPointerException` exception is created.
- Updating an enterprise bean custom finder method; the new SQL statement does not run.
- Changing schema mapping; the new SQL statement does not run.

In summary, if you add or update an enterprise bean that contains a custom finder method, you must stop and then restart the server.

Data access problems for Oracle data sources

This topic provides troubleshooting tips for accessing Oracle data sources.

What error do you see when you try to access your Oracle-based data source?

- "Invalid Oracle URL is specified" on page 23

- “DSRA0080E: Exception was received by the data store adapter. See original exception message: ORA-00600” when connecting to or using an Oracle data source”
- “DSRA8100E: Unable to get a {0} from the data source. Explanation: See the linkedException for more information.” on page 24
- “Error while trying to retrieve text for error” error when connecting to an Oracle data source” on page 24
- “Class loader errors occur when using an Oracle OCI driver as your JDBC provider” on page 24
- “java.lang.UnsatisfiedLinkError:” connecting to an Oracle data source” on page 24
- “Receive java.lang.NullPointerException errors when referencing Oracle classes, or “ internal error: oracle.jdbc.oci. OCIEnv” errors when connecting to an Oracle data source” on page 24
- “WSVR0016W: Class path entry for the Oracle JDBC Thin Driver has an invalid variable” on page 25
- “Transaction recovery failure (for XA data sources)” on page 25
- “Application server receives an error in the SystemOut.log file when it registers the Oracle JDBC Diagnosability MBean” on page 25

Invalid Oracle URL is specified

This error might be caused by an incorrectly specified URL on the URL property of the target data source.

Examine the URL property for the data source object in the administrative console to ensure that it is correct.

Examples of Oracle URLs:

- For the thin driver: jdbc:oracle:thin:@//hostname:1521/myDatabase
- For the thick (OCI) driver: jdbc:oracle:oci:@tnsname1

"DSRA0080E: Exception was received by the data store adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source

A possible reason for this exception is that the version of the Oracle JDBC driver being used is older than the Oracle database. It is possible that more than one version of the Oracle JDBC driver is configured on the WebSphere Application Server.

Examine the version of the JDBC driver. Sometimes you can determine the version by looking at the class path to determine what directory the driver is in.

If you cannot determine the version this way, use the following program to determine the version. Before running the program, set the class path to the location of your JDBC driver files.

```
import java.sql.*;
import oracle.jdbc.driver.*;
class JDBCVersion
{
    public static void main (String args[])
    throws SQLException
    {
        // Load the Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // Get a connection to a database
        Connection conn = DriverManager.getConnection
        ("jdbc:oracle:thin:@appalooosa:1521:app1","sys","change_on_install");
        // Create Oracle DatabaseMetaData object
        DatabaseMetaData meta = conn.getMetaData();
        // gets driver info:
        System.out.println("JDBC driver version is " + meta.getDriverVersion());
    }
}
```

If the driver and the database are at different versions, replace the JDBC driver with the correct version. If multiple drivers are configured, remove any that occur at the incorrect level.

DSRA8100E: Unable to get a {0} from the data source. Explanation: See the linkedException for more information.

When using an Oracle thin driver, Oracle creates a `java.sql.SQLException: invalid arguments in call` error if no user name or password is specified when getting a connection. If you see this error while running WebSphere Application Server, the alias is not set.

To remove the exception, define the alias on the data source.

"Error while trying to retrieve text for error" error when connecting to an Oracle data source

The likely cause of this error is that the Oracle OCI driver is being used with an `ORACLE_HOME` property that is either not set or is set incorrectly.

To correct the error, examine the user profile that WebSphere Application Server is running under to verify that the `$ORACLE_HOME` environment variable is set correctly.

Class loader errors occur when using an Oracle OCI driver as your JDBC provider

When you configure an Oracle OCI driver as your JDBC provider, you must specify the path to where the native libraries are stored. If you did not specify a native library path, the first time you try to connect using this provider, class loader errors occur.

To correct this problem, in the administrative console, click **Resources > JDBC > JDBC Providers**, select the Oracle OCI driver, and then specify the path to the native libraries in the **Native library path** field.

"java.lang.UnsatisfiedLinkError:" connecting to an Oracle data source

The environment variable `LIBPATH` might not be set or is set incorrectly, if your data source creates an `UnsatisfiedLinkError` error, and the full exception indicates that the problem is related to an Oracle module.

To correct the problem:

1. Examine the user profile under which the WebSphere Application Server is running to verify that the `LIBPATH` environment variable includes Oracle libraries.
2. Configure the Java `LIBPATH` (`java.library.path`) property with the `domain_region_libpath` environment variable, like `control_region_libpath`, `server_region_libpath`, `adjunct_region_libpath`. See the topic on changing the values of variables referenced in BBOM0001I messages for instructions on how to set the region `LIBPATH` variables.

Receive java.lang.NullPointerException errors when referencing Oracle classes, or "internal error: oracle.jdbc.oci. OCIEnv" errors when connecting to an Oracle data source

The problem might be that the OCI driver is being used on an AIX machine, the `LIBPATH` is set correctly, but the `ORACLE_HOME` environment variable is not set or is set incorrectly. You can encounter an exception similar to either of the following when your application attempts to connect to an Oracle data source:

To correct the problem, examine the user profile that WebSphere Application Server is running under to verify that it has the `$ORACLE_HOME` environment variable set correctly, and that the `$LIBPATH` includes `$ORACLE_HOME/lib`.

WSVR0016W: Class path entry for the Oracle JDBC Thin Driver has an invalid variable

This error occurs when there is no environment variable defined for the property, ORACLE_JDBC_DRIVER_PATH.

Verify this problem in the administrative console. Go to **Environment > Manage WebSphere Variables** to verify that the variable *ORACLE_JDBC_DRIVER_PATH* is defined.

To correct the problem, click **New** and define the variable. For example, name: ORACLE_JDBC_DRIVER_PATH, value : c:\oracle\jdbc\lib. Use a value that names the directory in your operating system that contains the ojdbc6.jar file (or the ojdbc6_g.jar file to enable Oracle trace.)

Transaction recovery failure (for XA data sources)

Problem

When WebSphere Application Server attempts to recover Oracle database transactions, the transaction service issues the following exception:

```
WTRN0037W: The transaction service encountered an error on an xa_recover operation.
The resource was com.ibm.ws.rsadapter.spi.WSRdbXaResourceImpl@1114a62.
The error code was XAER_RMERR. The exception stack trace follows:
javax.transaction.xa.XAException
at oracle.jdbc.xa.OracleXAResource.recover(OracleXAResource.java:726)
at com.ibm.ws.rsadapter.spi.WSRdbXaResourceImpl.recover(WSRdbXaResourceImpl.java:954)
at com.ibm.ws.Transaction.JTA.XARminst.recover(XARminst.java:137)
at com.ibm.ws.Transaction.JTA.XARecoveryData.recover(XARecoveryData.java:609)
at com.ibm.ws.Transaction.JTA.PartnerLogTable.recover(PartnerLogTable.java:511)
at com.ibm.ws.Transaction.JTA.RecoveryManager.resync(RecoveryManager.java:1784)
at com.ibm.ws.Transaction.JTA.RecoveryManager.run(RecoveryManager.java:2241)
```

Cause

Oracle requires services such as the WebSphere Application Server transaction service to have special permissions for performing transaction recoveries.

Solution

As user **SYS**, run the following commands on your Oracle server:

```
grant select on pending_trans$ to public;
grant select on dba_2pc_pending to public;
grant select on dba_pending_transactions to public;
grant execute on dbms_system to <user>;
```

User is a user ID in the application server that is authorized to perform transaction recovery for the XA data source. If you have not authorized any user IDs to perform transaction recovery, the application server uses the login alias for the data source as the user ID.

This problem is mentioned under Oracle bug: 3979190. Running the preceding commands solves the problem.

Application server receives an error in the SystemOut.log file when it registers the Oracle JDBC Diagnosability MBean

The following error is displayed in the SystemOut.log file when WebSphere Application Server attempts to connect to an Oracle database:

```
E Error while registering Oracle JDBC Diagnosability MBean. javax.management.MalformedObjectNameException: Invalid character
```

This error occurs during the initial connection to an Oracle database because the diagnosis for the MBean is not properly initialized.

You can ignore this error. However, you can apply patch number 6362104 from Oracle to prevent future occurrences of this error. Check with Oracle if you have any other patches applied, as this patch might not work with other patches.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Data access problems for DB2 databases

This article provides troubleshooting tips for accessing DB2 databases.

What kind of problem are you having accessing your DB2 database?

- “Kerberos login error occurs while connecting to the database” on page 27
- “SQL0567N "DB2ADMIN " is not a valid authorization ID. SQLSTATE=42602” on page 28
- “SQL0805N Package package-name was not found” on page 28
- “SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002” on page 28
- “SQL30082N Attempt to establish connection failed with security reason "17" ("UNSUPPORTED FUNCTION") SQLSTATE=08001” on page 28
- “SQLException, with ErrorCode -99,999 and SQLState 58004, with Java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source” on page 29
- “Error message java.lang.reflect.InvocationTargetException: com.ibm.ws.exception.WsException: DSRA0023E: The DataSource implementation class "COM.ibm.db2.jdbc.DB2XADDataSource" could not be found. when trying to access a DB2 database” on page 29
- “CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800” on page 30
- “COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001” on page 30
- ““COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])” on page 31
- “Error 10500 (E_Fatal) returns when an attempt is made to either publish or inquire on an entity exceeding 255 characters in one of its fields.” on page 31
- “ java.sql.SQLException: Failure in loading T2 native library db2jcct2 DSRA0010E: SQL State = null, Error Code = -99,999 ” on page 31
- Lock contention exception occurs in database when data source implementation type is XA
- ““DSRA8050W: Unable to find the DataStoreHelper class specified" exception occurs when trying to use a DB2 Universal Datasource in a mixed release cell.” on page 32
- “Receive "'SYSTEM' is not a valid authorization ID" message when trying to access DB2 on a Windows machine where WebSphere Application Server is also installed.” on page 33
- “XAException: XAER_NOTA on XA prepare call in DB2 Universal JDBC Driver type 4 after one phase transaction rollback” on page 33
- “java.rmi.MarshalException logged for application client due to incompatibility of JDBC driver file versions” on page 34
- “Database failure triggers problematic -99999 exception for applications that use DB2 Universal Driver type 4” on page 34
- “Cannot access DB2 on Linux when using the DB2 Universal JDBC Driver” on page 35
- “Illegal conversion occurs on any VARCHAR FOR BIT DATA column in a container-managed persistent bean” on page 36

Kerberos login error occurs while connecting to the database

If you are using the following list of scenarios, a kerberos log-in error occurs while trying to connect to the database.

1. The injection statements in an Enterprise JavaBeans (EJB) 3.0 bean are at the attribute level.
2. The persistence.xml file does not call out metadata information regarding the database to which you have a connected.
3. The component managed alias on the data source is not valid.
4. The database is set up to use a one-off security mechanism, for example, a mechanism other than the typical user ID and password mechanism, such as kerberos.

During the injection of an EJB 3.0 bean, the Java Persistence API (JPA) persistence.xml file attempts to connect to the database to lookup metadata. To make the connection, a Java™ 2 Connector (J2C) requests a Subject from the security context. In this scenario, the security collaboration API that is called to put information into the security context was not called and the subject is returned without containing GSSCredentials.

The call should have taken place in the EJB container, but the EJB container did not call the collaboration API because the security APIs require a fully constructed bean that did not exist. Also, the EJB 3.0 specification explicitly states that the bean construction should not be done in a defined security context. Because of the design of the security API and the direction of the EJB 3.0 specification, the EJB container did not call the security collaboration APIs, therefore, the security context did not know to set up the Subject and there were no GSSCredentials.

The absence of GSSCredentials causes the Relational Resource Adapter (RRA) implementation to use the wrong code path; instead of telling DB2 to connect using the GSSCredential, it uses the identity associated with the component managed alias. The result is that the component managed alias is configured to be an identity that is not known to kerberos, therefore, when the DB2 driver relayed this identity to the database, an error occurred.

The following code is an example of the error:

```
Caused by: javax.security.auth.login.FailedLoginException: Login error: com.ibm.security.krb5.KrbException, status code:
message: Client not found in Kerberos database
at com.ibm.security.jgss.i18n.I18NException.throwFailedLoginException(I18NException.java:25)
at com.ibm.security.auth.module.Krb5LoginModule.b(Krb5LoginModule.java:733)
at com.ibm.security.auth.module.Krb5LoginModule.c(Krb5LoginModule.java:610)
at com.ibm.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.java:433)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:45)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37)
at java.lang.reflect.Method.invoke(Method.java:599)
at javax.security.auth.login.LoginContext.invoke(LoginContext.java:795)
at javax.security.auth.login.LoginContext.access$000(LoginContext.java:209)
at javax.security.auth.login.LoginContext$4.run(LoginContext.java:709)
at java.security.AccessController.doPrivileged(AccessController.java:251)
at javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:706)
at javax.security.auth.login.LoginContext.login(LoginContext.java:603)
at com.ibm.db2.jcc.a.ud.a(ud.java:25)
```

To prevent this error, follow these recommendations:

- Ensure that the injection statements are at "class level."
This prevents you from attempting to do injection when the bean is created. The persistence context items are still stored in the Java Naming and Directory Interface (JNDI), but they are not placed into local variables on the bean object. The bean method looks up these persistence context items from the JNDI and by that time, the bean should be running in the correct security context because the EJB container has called the security collaboration APIs.
- Have the persistence.xml file call out the database metadata information.

If the persistence.xml file calls out the needed database metadata information, there is not a need for the JPA to connect to the database to retrieve it. This means that a connection is not made until the bean method is accessed. By this step, the security context is correctly set up and the connection works.

- Validate the component managed alias on the data source.

When the JPA attempts to connect to the database for purposes of gathering metadata, the driver falls back to using the component managed alias that is set on the data source because the security context has not been configured. If you make sure to validate the component managed alias, the connection is successful.

- Do not use a one-off security mechanism for database security.

This is not an option if you want to use a one-off security mechanism, such as kerberos. If the database is not configured to accept the standard user ID and password authentication this problem does not occur. The problem only occurs if when your system requires special security configuration.

SQL0567N "DB2ADMIN " is not a valid authorization ID. SQLSTATE=42602

If you encounter this error when attempting to access a DB2 Universal Database™ (UDB):

1. Verify that your user name and password in the data source properties page in the administrative console are correct.
2. Ensure that the user ID and password do not contain blank characters before, in between, or after.

SQL0805N Package *package-name* was not found

Possible reasons for these exceptions:

- If the package name is NULLID.SQLLC300, see SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002. for the reason.
- You are attempting to use an XA-enabled JDBC driver on a DB2 database that is not XA-ready.

To correct the problem on a DB2 Universal Database (UDB), run this one-time procedure, using the db2cmd interface while connected to the database in question:

1. **DB2 bind @db2ubind.lst blocking all grant public**
2. **DB2 bind @db2cli.lst blocking all grant public**

The db2ubind.lst and db2cli.lst files are in the bnd directory of your DB2 installation root. Run the commands from that directory.

SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002

This error can occur because:

- The underlying database was dropped and recreated.
- DB2 was upgraded and its packages are not rebound correctly.

To resolve this problem, rebound the DB2 packages by running the db2cli.lst script found in the bnd directory. For example: db2>@db2cli.lst.

SQL30082N Attempt to establish connection failed with security reason "17" ("UNSUPPORTED FUNCTION") SQLSTATE=08001

This error can occur when the security mechanism specified by the client is not valid for this server. Some typical examples:

- The client sent a new password value to a server that does not support the change password function.
- The client sent SERVER_ENCRYPT authentication information to a server that does not support password encryption.
- The client sent a userid, but no password, to a server that does not support authentication by userid only.

- The client has not specified an authentication type, and the server has not responded with a supported type. This can include the server returning multiple types from which the client is unable to choose.

To resolve this problem, ensure that your client and server use the same security mechanism. For example, if this is an error on your data source, verify that you have assigned a user id and password or authentication alias.

SQLException, with ErrorCode -99,999 and SQLState 58004, with Java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source

An unexpected system failure usually occurs when running in XA mode (two-phase commit). Among the many possible causes are:

- An invalid username or password was provided.
- The database name is incorrect.
- Some DB2 packages are corrupted.

To determine whether you have a user name or password problem, look in the db2diag.log file to view the actual error message and SQL code. A message like the following example, with an SQLCODE of -1403, indicates an invalid user ID or password:

```
2002-07-26-14.19.32.762905 Instance:db2inst1 Node:000
PID:9086(java) Appid:*LOCAL.db2inst1.020726191932
XA DTP Support sqlxa_open Probe:101
DIA4701E Database "POLICY2" could not be opened
for distributed transaction processing.
String Title: XA Interface SQLCA PID:9086 Node:000
SQLCODE = -1403
```

To resolve these problems:

1. Correct your user name and password. If you specify your password on the GUI for the data source, ensure that the user name and password you specify on the bean are correct. The user name and password you specify on the bean overwrite whatever you specify when creating the data source.
2. Use the correct database name.
3. Rebind the packages (in the bnd directory) as follows:


```
db2connect to dbname
c:\SQLLIB\bnd>DB2 bind @db2ubind.lst blocking all grant public
c:\SQLLIB\bnd>DB2 bind @db2cli.lst blocking all grant public
```
4. Ensure that the \WebSphere\AppServer\properties\wsj2cdpm.properties file has the right user ID and password.

Error message java.lang.reflect.InvocationTargetException: com.ibm.ws.exception.WsException: DSRA0023E: The DataSource implementation class "COM.ibm.db2.jdbc.DB2XADataSource" could not be found. when trying to access a DB2 database

One possible reason for this exception is that a user is attempting to use a JDBC 2.0 DataSource, but DB2 is not JDBC 2.0-enabled. This situation frequently happens with new installations of DB2 because DB2 provides separate drivers for JDBC 1.X and 2.0, with the same physical file name. By default, the JDBC 1.X driver is on the class path.

To confirm this problem:

- On Windows systems, look for the inuse file in the java12 directory in your DB2 installation root. If the file missing, you are using the JDBC 1.x driver.
- On operating systems such as AIX or Linux, check the class path for your data source. If the class path does not point to the db2java.zip file in the java12 directory, you are using the JDBC 1.x driver.

To correct this problem:

- On Windows systems, stop DB2. Run the usejdbc2.bat file from the java12 directory in your DB2 installation root. Run this file from a command line to verify that it completes successfully.
- On operating systems such as AIX or Linux, change the class path for your data source to point to the db2java.zip file in the java12 directory of your DB2 installation root.

CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800

If you encounter this error when attempting to access a DB2 Universal Database (UDB) data source:

1. On the data source properties page in the administrative console, verify that the correct database name is specified on the data source.
2. On the custom properties page, check your user name and password custom properties. Verify that they are correct.
3. Ensure the user ID and password do not contain any blank characters, before, in between, or after.
4. Check that the WAS.policy file exists for the application, for example, D:\WebSphere\AppServer\installedApps\markSection.ear\META-INF\was.policy.
5. View the entire exception listing for an underlying SQL error, and look it up using the DBM vendor message reference.

If you encounter this error while running DB2 on Red Hat Linux, the **max queues system wide** parameter is too low to support DB2 while it acquires the necessary resources to complete the transaction. When this problem exists, the exceptions J2CA0046E and DSRA0010E can precede the exception DSRA8100E.

To correct this problem, edit the /proc/sys/kernal/msgmni file to increase the value of the **max queues system wide** parameter to a value greater than 128.

COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001

This problem is probably an application-caused DB2 deadlock, particularly if you see an error similar to the following when accessing a DB2 data source:

```
ERROR CODE: -911
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N
The current transaction has been rolled back because of a deadlock or timeout.
Reason code "2". SQLSTATE=40001
```

To diagnose the problem:

1. Execute these DB2 commands:
 - a. db2 update monitor switches using LOCK ON
 - b. db2 get snapshot for LOCKS on dbName >

The *directory_name\lock_snapshot.log* now has the DB2 lock information.
2. Turn off the lock monitor by running: db2 update monitor switches using LOCK OFF

To verify that you have a deadlock:

1. Look for an application handle that has a lock-wait status, and then look for the ID of the agent holding lock to verify the ID of the agent.
2. Go to that handle to verify it has a lock-wait status, and the ID of the agent holding the lock for it. If it is the same agent ID as the previous one, then you know that you have a circular lock (deadlock).

To resolve the problem:

1. Examine your application and use a less restrictive isolation level if no concurrency access is needed.
2. Use caution when changing the **accessIntent** value to move to a lower isolation level. This change can result in data integrity problems.

3. For DB2/UDB Version 7.2 and earlier releases, you can set the DB2_RR_TO_RS flag from the DB2 command line window to eliminate unnecessary deadlocks, such as when the accessIntent defined on the bean method is too restrictive, for example, PessimisticUpdate. The DB@_RR_TO_RS setting has two impacts:

- If RR is your chosen isolation level, it is effectively downgraded to RS.
- If you choose another isolation level, and the DB2_RR_TO_RS setting is on, a scan skips over rows that are deleted but not committed, even though the row might qualify for the scan. The skipping behavior affects the RR, Read Stability (RS), and Cursor Stability (CS) isolation levels.

For example, consider the scenario where transaction A deletes the row with column1=10 and transaction B does a scan where column1>8 and column1<12. With DB2_RR_TO_RS off, transaction B waits for transaction A to commit or rollback. If transaction A rolls back, the row with column1=10 is included in the result set of the transaction B query. With DB2_RR_TO_RS on, transaction B does not wait for transaction A to commit or rollback. Transaction B immediately receives query results that do not include the deleted row. Setting DB2_RR_TO_RS effectively changes locking behavior, thus avoiding deadlocks.

"COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])"

This error is denoted by message DSRA8040I: Failed to connect to the DataSource.

This error usually occurs when the class path of the DB2 JDBC driver is set correctly to `${DB2_JDBC_DRIVER_PATH}/db2java.zip` but the environment variable DB2_JDBC_DRIVER_PATH is not set.

This error can also occur if you are using DB2 Version 7.1 or 7.2 and you have not yet run `usejdbc2`. This might be the problem if your path is correct but you still receive this error.

To confirm this problem:

1. Go to the **Manage WebSphere Variables** panel.
2. Select **Environment** to verify that there is no entry for the variable DB2_JDBC_DRIVER_PATH.

To correct this problem: Add the variable DB2_JDBC_DRIVER_PATH with **value** equal to the directory path containing the db2java.zip file.

Error 10500 (E_Fatal) returns when an attempt is made to either publish or inquire on an entity exceeding 255 characters in one of its fields.

This error usually occurs when you make an attempt to either publish or inquire on an entity exceeding 255 characters in one of its fields. This is less obvious when using non-English characters, as the real limit is reached before 255 visual characters are displayed.

To correct this problem: Accept this as a limitation in using DB2 Version 7 on z/OS. Do not exceed the 255 character limit.

java.sql.SQLException: Failure in loading T2 native library db2jct2 DSRA0010E: SQL State = null, Error Code = -99,999

The *Failure in loading* message indicates one of the following problems

- A machine was not rebooted after installing DB2. Reboot the machine getting the error and try it again.
- A database operation was performed that uses a different scope than the configured data source. For example, the testConnection command runs at the node level for a data source that exists on the server level. Source the db2profile script on the machine and ensure that the environment contains pointers to the DB2 native libraries. The db2profile script exists in the root directory of the DB2 user ID. For more information on the testConnection command, see Test connection service.

- The DB2 context is not getting set up correctly for the user running WebSphere Application Server. Source the db2profile script on the machine and ensure that the environment contains pointers to the DB2 native libraries.

Lock contention exception occurs in database when data source implementation type is XA

Note: Because a lock contention exception can be caused by many factors, consider the following explanation and recommended response as a strategy for eliminating the possible reasons for your lock contention problem.

Symptom	A lock contention exception occurs in a DB2 database that your application accesses through a data source of implementation type XA.
Problem	Your application is trying to access database records that are locked by an XA transaction that is in ended (e) state, but cannot be prepared by the transaction manager.
Description	An XA transaction to DB2 that ends, but cannot be prepared, is in ended (e) state. Because it is <i>not</i> considered to be <i>in doubt</i> , the transaction manager cannot recover this transaction. DB2 does not return it in the list of in doubt transactions. DB2 also does not roll the transaction back immediately; it waits until all connections to the database are released. During this period of inaction, the transaction continues to hold locks on the database. If your application server does not disconnect all connections from the database to allow rollback, the ended transaction persists in locking the same database records. If your application attempts to access these locked records, a lock contention exception occurs in DB2.
Recommended response	DB2 Version 8.2 is shipped with a sample application that connects to a defined DB2 server and uses the available DB2 APIs to obtain a list of these particular ended transactions. The application offers a configuration setting that enables you to designate an amount of time after which the application rolls these transactions back. Locate the sample application in the <code>sql11ib/samples/db2xamon.c</code> directory of DB2 Version 8.2 and run it.

"DSRA8050W: Unable to find the DataStoreHelper class specified" exception occurs when trying to use a DB2 Universal Datasource in a mixed release cell.

This error usually occurs when you are using WebSphere Application Server Version 6.0 or above in conjunction with a previous version and attempt to create a DB2 Universal Datasource on the previous version.

This can happen because the DB2 Universal Datasource was not available on Version 5 and previous versions, but the Version 6 administrative console allows you to build one.

To correct this problem: create the datasource on Version 6.0 or later.

Receive "'SYSTEM' is not a valid authorization ID" message when trying to access DB2 on a Windows machine where WebSphere Application Server is also installed.

Table 3. Error message description and resolution. You have two options to solve the problem.

Symptom	<p>For a WebSphere Application Server on Windows installation that uses DB2 as the backend, you see the following exception in the JVM log:</p> <pre>java.sql.SQLException: [IBM][CLI Driver] SQL0567N "SYSTEM" is not a valid authorization ID. SQLSTATE=42602 DSRA0010E: SQL State = 42602, Error Code = -567 at COM.ibm.db2.jdbc.app.SQLExceptionGenerator.throw_SQLException (Unknown Source) at COM.ibm.db2.jdbc.app.SQLExceptionGenerator.check_return_code (Unknown Source) at COM.ibm.db2.jdbc.app.DB2Connection.connect(Unknown Source) at COM.ibm.db2.jdbc.app.DB2Connection.<init>(Unknown Source) at COM.ibm.db2.jdbc.app.DB2ReusableConnection.<init>(Unknown Source) at COM.ibm.db2.jdbc.DB2PooledConnection.getConnection(Unknown Source) at com.ibm.ws.rsadapter.spi.WSRdbDataSource.getConnection (WSRdbDataSource.java:1035) at com.ibm.ws.rsadapter.spi.WSManagedConnectionFactoryImpl. createManagedConnection(WSManagedConnectionFactoryImpl.java:937) at com.ibm.ejs.j2c.poolmanager.FreePool. createManagedConnectionWithMCWrapper(FreePool.java:1502)</pre>
Problem	<p>This exception occurs for configurations in which WebSphere Application Server is a client to the DB2 server. The underlying problem is an authorization conflict between WebSphere Application Server on Windows and DB2 that arises when an application attempts to connect to DB2 without providing a user ID and a password.</p>
Description	<p>When a DB2 client and the DB2 database run on the same machine, DB2 allows the client to connect without a user ID and password. The connection is made under the credentials of the user that owns the client process: in this case, the application server JVM. However, if WebSphere Application Server runs as a Windows service, and the "Log on as" option is set to "Local System Account", the application server JVM is categorized as a subcomponent of a special Windows user called SYSTEM. This user is not allowed to connect to DB2, resulting in the previously shown exception.</p>
Recommended response	<p>You have two options:</p> <ul style="list-style-type: none"> • Modify the WebSphere Application Server service to use a Log on as option of This account, and provide an account with permission to connect to DB2. Or • Configure your application server to provide credentials on the DB2 connection by using container-managed or component-managed authentication.

XAException: XAER_NOTA on XA prepare call in DB2 Universal JDBC Driver type 4 after one phase transaction rollback

Symptom

For applications that use the DB2 Universal JDBC Driver type 4 XA available with DB2 v8.2, a connection might fail and trigger an XAER_NOTA XAException error. The following code block is an example of this exception:

```
J2CA0027E: An exception occurred while invoking prepare
on an XA Resource Adapter from dataSource jdbc/SDOSVT,
within transaction ID {XidImpl: formatId(57415344),
gtrid_length(36), bqual_length(54),
data(000000ff51913982000000001000000296cac5c42fe3c6838631cbaafc8b5a9253b846544
000000ff51913982000000001000000296cac5c42fe3c6838631cbaafc8b5a9253b84654400000000
10000000000000000000000000002)}:
javax.transaction.xa.XAException: XAER_NOTA
at com.ibm.db2.jcc.a.xb.a(xb.java:1682)
at com.ibm.db2.jcc.a.xb.a(xb.java:841)
```

```
at com.ibm.db2.jcc.a.xb.prepare(xb.java:812)
at com.ibm.ws.rsadapter.spi.WSRdbXaResourceImpl.prepare
(WSRdbXaResourceImpl.java:837)
...
```

Problem

If a DB2 Universal JDBC Driver type 4 XA connection is used in a single-phase transaction, such as a local transaction with autocommit set to false, and that single-phase transaction is rolled back, the next use of the connection in a two-phase transaction fails on the prepare call.

A problem in the DB2 Universal JDBC Driver type 4 XA support causes the XA prepare call to fail. This problem does not occur if the single-phase transaction is committed, and it does not occur when using the DB2 Universal JDBC Driver in type 2 mode.

Solution

Upgrade to DB2 Version 8.2 Fix Pack 1, which is equivalent to Version 8.1 Fix Pack 8. The Universal JDBC Driver XA that is available with these releases solves the previously described issue for type 4 mode.

java.rmi.MarshalException logged for application client due to incompatibility of JDBC driver file versions

Symptom

For an application that includes a application client, the following error message is displayed in the client log file of the application server:

```
java.rmi.MarshalException: CORBA MARSHAL
0x4942f89a No; nested exception is:
org.omg.CORBA.MARSHAL: Unable to read value from
underlying bridge : Mismatched serialization
UIDs : Source (Rep.
IDRMI:com.ibm.db2.jcc.c.SqlException:63EEE52211DCD763:82CE0C0DA2B0A000)
= 82CE0C0DA2B0A000 whereas Target (Rep. ID
RMI:com.ibm.db2.jcc.c.SqlException:63EEE52211DCD763:91C6171BC645E41B)
= 91C6171BC645E41B vmcid: 0x4942f000 minor code:
2202 completed: No
```

Problem

The db2jcc.jar files on the application client machine and on your application server are from versions of DB2 that are not compatible with each other, or are not compatible with the version of DB2 that functions as the datastore.

Solution

Check the db2jcc.jar files on the application client machine, your application server, and your DB2 server. On the client machine and the application server, install files of the same version that is compatible with the DB2 server.

Database failure triggers problematic -99999 exception for applications that use DB2 Universal Driver type 4

Symptom

If you use the DB2 Universal Driver type 4 for access to DB2 or Cloudscape Network Server, and your database fails, the database server issues a generic -99999 exception in response to every JDBC getConnection request. This exception, which is exemplified in the following code excerpt, can cause unexpected behavior in your applications.

```
java.sql.SQLException: IO Exception opening socket to
server bs8.rchland.ibm.com on port 1527.
The DB2 Server may be down.DSRA0010E: SQL State = null,
Error Code = -99,999DSRA0010E: SQL State = null,
Error Code = -99,999
at com.ibm.db2.jcc.b.a.<init>(a.java:125)
at com.ibm.db2.jcc.b.b.a(b.java:1011)
at com.ibm.db2.jcc.c.l.<init>(l.java:197)
at com.ibm.db2.jcc.b.b.<init>(b.java:258)
at com.ibm.db2.jcc.DB2PooledConnection.
<init>(DB2PooledConnection.java:44)
at com.ibm.db2.jcc.DB2ConnectionPoolDataSource.getPooledConnectionX
(DB2ConnectionPoolDataSource.java:80)
at com.ibm.db2.jcc.DB2ConnectionPoolDataSource.getPooledConnection
(DB2ConnectionPoolDataSource.java:45)
at com.ibm.ws.rsadapter.DSConfigurationHelper$1.run
(DSConfigurationHelper.java:945)
```

Problem

When running in type 4 mode, some versions of the DB2 Universal Driver trigger a generic exception for database failure rather than a specific error code that WebSphere Application Server can map to a stale connection exception. This problem occurs with versions of the driver that are associated with DB2 8.1 Fix Pack 6 or Fix Pack 7, and DB2 8.2.

Solution

Upgrade to DB2 Version 8.2 Fix Pack 1, equivalent to Version 8.1 Fix Pack 8, which provides a valid error code in the previously described scenario. WebSphere Application Server maps this error code to a StaleConnectionException, as expected.

Cannot access DB2 on Linux when using the DB2 Universal JDBC Driver

Symptom

In the WebSphere Application Server on Linux environment, applications that use the DB2 Universal JDBC Driver to access DB2 on Linux might not connect with the database. The database server can issue the following exceptions to the application server error log:

- java.security.AccessControlException: Access denied
(java.lang.RuntimePermission accessClassInPackage.sun.io)
- *If you run 64-bit Linux:*
com.ibm.db2.jcc.b.SQLException: Failure in loading T2 native library db2jcct2

Problem

The process for configuring DB2 on Linux to work with the Universal JDBC Driver is not complete.

Solution

- Verify that the setup requirements for the Java™ SDK 1.4.2 for the Linux platform are complete.
- Configure your development environment for building Java applications on Linux with DB2 JDBC support. See the application development topics for DB2 for more information.
- If you run DB2 on the Linux/IA64 platform and are using DB2 v8.1 Fix Pack 7A, perform the additional step that is described in the technote on DB2 UDB Version 8 FixPak 7a for Linux on IA64 reporting a

missing libdb.so.3 library. This step is necessary only for Fix Pack 7A. This step is not necessary for DB2 v8.1 Fix Pack 7 or earlier versions of DB2; this step is not necessary for DB2 v8.1 Fix Pack 8 or later versions of DB2.

Illegal conversion occurs on any VARCHAR FOR BIT DATA column in a container-managed persistent bean

When enterprise beans with container-managed persistent (CMP) types that have any VARCHAR FOR BIT DATA columns defined on a DB2 table are deployed in the DB2 universal JDBC type 4 driver to persist the data, an SQLException of illegal conversion is thrown at run time. This exception only occurs when you use the DB2 universal JDBC type 4 driver and with the deferPrepares property being set to true. When the deferPrepares property is set to true, the DB2 universal JDBC type 4 driver uses the standard JDBC data mapping.

Currently, the generated deployed code does not follow the standard JDBC specification mapping. The failure at execution time is because of a problem in the tool that prepared the enterprise beans for execution.

To avoid receiving this exception, choose one of the following options:

- Set the deferPrepares property to false in the data source configuration.
- Do not use the DB2 universal JDBC type 4 driver if your table has any VARCHAR FOR BIT DATA or LONG VARCHAR FOR BIT DATA columns. Refer to DB2 V8.1 readme for more details.

Data access problems for Microsoft SQL Server data sources

This topic provides troubleshooting tips for accessing Microsoft SQL Server data sources.

What problem are you having accessing your Microsoft SQL Server database?

- “Hang in Microsoft SQL Server JDBC Driver V2.0 after connection error”
- “ERROR CODE: 20001 and SQL STATE: HY000 accessing SQLServer database”
- “Application fails with message stating "Cannot find stored procedure..." accessing a Microsoft SQL Server database” on page 37
- “ERROR CODE: SQL5042 when you run a Java application” on page 37

Hang in Microsoft SQL Server JDBC Driver V2.0 after connection error

If you are using version 2.0 of the Microsoft SQL Server JDBC Driver (other versions do not have the issue), you might experience a hang after a connection error occurs. The following test fix from Microsoft fixes this issue: <http://support.microsoft.com/kb/977924>

ERROR CODE: 20001 and SQL STATE: HY000 accessing SQLServer database

The problem might be that the distributed transaction coordinator service is not started. Look for an error similar to the following example when attempting to access a Microsoft SQL Server database:

```
ERROR CODE: 20001
SQL STATE: HY000
java.sql.SQLException: [Microsoft][SQLServer JDBC Driver]
[SQLServer]xa_open (0) returns -3
at com.microsoft.jdbc.base.BaseExceptions.createException(Unknown Source) ...
at com.microsoft.jdbcx.sqlserver.SQLServerDataSource.getXAConnection
(Unknown Source) ...
```

To confirm this problem:

1. Go to the Windows **Control Panel > Services**. Or, click **Control Panel > Administrative Tools > Services**.
2. Verify whether the service **Distributed Transaction Coordinator** or **DTC** is started.
3. If not, start the **Distributed Transaction Coordinator** service.

Application fails with message stating "Cannot find stored procedure..." accessing a Microsoft SQL Server database

This error can occur because the stored procedures for the Java Transaction API (JTA) feature are not installed on the Microsoft SQL Server.

To resolve the problem, repeat the installation for the stored procedures for the JTA feature, according to the JDBC driver installation guide.

ERROR CODE: SQL5042 when you run a Java application

This error can occur when you configure your application to run in the following manner:

1. You use a type 2 (application) driver running on the gateway to the OS 390
2. Your application is an XA application.

OS 390 does not use XA, but uses SPM. To resolve the problem:

1. Check your dbm cfg to see that the SPM is not started on the gateway.
2. Assign a port and set the *db2comm* variable to TCPIP.
3. Update the dbm cfg value *SPM_NAME* to use your machine name.
4. Start the SPM on the gateway.

Data access problems for Apache Derby databases

This topic provides troubleshooting tips for accessing Apache Derby databases.

What problem are you having accessing your Apache Derby database?

- "Unexpected IOException wrapped in SQLException, accessing Apache Derby database"
- "The "select for update" operation causes table lock and deadlock when accessing Apache Derby"
- "Error "The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Apache Derby databases"" on page 38
- "Running an application causes a runtime exception which produces an unreadable message" on page 38

Attention: Apache Derby errorCodes 2000, 3000, and 4000, indicate levels of severity, not specific error conditions. In diagnosing Apache Derby problems, pay attention to the given *sqlState* value.

Unexpected IOException wrapped in SQLException, accessing Apache Derby database

This problem can occur because Apache Derby databases use many files. Some operating systems, such as the Solaris Operating Environment, limit the number of files an application can open at one time. If the default is a low number, such as 64, you can get this exception.

If you can configure the number of file descriptors on your operating system, you can correct the problem by setting the number to a high value, such as 1024.

The "select for update" operation causes table lock and deadlock when accessing Apache Derby

If a select for update operation on one row locks the entire table, which creates a deadlock condition, there might be undefined indexes on that table. The lack of an index on the columns you use in the where clause can cause Apache Derby to create a table lock rather than a row level lock.

To resolve this problem, create an index on the affected table.

Error "The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Apache Derby databases"

At the client run time, an error similar to the following occurs:

```
The version of the IBM Universal JDBC driver in use is not
licensed for connectivity to Apache Derby databases. To connect
to this DB2 server, please obtain a licensed copy of the IBM DB2
Universal Driver for JDBC and SQLJ. An appropriate license file
db2jcc_license_*.jar for this target platform must be installed to
the application classpath. Connectivity to Apache Derby databases is
enabled by any of the following license files:
{ db2jcc_license_c.jar, b2jcc_license_cu.jar, db2jcc_license_cisuz.jar }
```

The problem occurs because an incorrect JDBC driver Java archive (JAR) file name is specified in the class path for JDBC provider. For example, the JAR file name might have an extra '_', as follows:

```
${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license__cu.jar
```

To resolve the problem:

1. Correct the UNIVERSAL_JDBC_DRIVER_PATH JAR file name in the JACL script.
2. Restart the cluster.
3. Rerun the client.

Running an application causes a runtime exception which produces an unreadable message

At client run time, you might receive a message similar to the following: Caused by:
com.ibm.db2.jcc.a.SqlException: DB2 SQL error: SQLCODE: -1, SQLSTATE: 42X05, SQLERRMC:
ANNUITYHOLDER20^T42X05

The problem occurs because the property *retrieveMessagesFromServerOnGetMessage*, which is required by WebSphere Application Server, has not been set.

To resolve the problem, on the administrative console

1. Click **Resources > JDBC Providers**.
2. Click an Apache Derby provider
3. Scroll down and click **Data Sources**.
4. Select your data source or add a new one.
5. Select **Custom Properties**.
6. If the property, *retrieveMessagesFromServerOnGetMessage* exists, set its value to true. If the property does not exist, select New and add the property, *retrieveMessagesFromServerOnGetMessage* with a value true.
7. Rerun the client.

The SystemOut.log now generates readable messages so that you can resolve the underlying problem.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Data access problems for Sybase data sources

This article provides troubleshooting tips for accessing Sybase data sources.

What kind of problem are you having accessing your Sybase database?

- "Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode" error.
- "JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server."
- A container managed persistence (CMP) enterprise bean is causing exceptions.
- "Sybase JDBC data source fails with "Incorrect URL format" exception in IPv6 environment" on page 40
- "Executing the DatabaseMetaData.getBestRowIdentifier() method in an XA transaction causes errors" on page 40
- "Sybase requirements for using the escapes and DatabaseMetaData methods" on page 41
- "Database deadlocks and XA_PROTO errors occur when using Sybase" on page 41
- "Executing a stored procedure containing a SELECT INTO command causes exception" on page 42
- "Error is incorrectly reported about IMAGE to VARBINARY conversion" on page 42
- "JDBC 1.0 standard methods are not implemented and generate a SQL exception when used" on page 42
- "Sybase transaction manager fails after trying to alleviate a deadlock error" on page 42
- "Starting an XA transaction when the autoCommit value of the connection is false causes error" on page 42
- "Sybase does not throw an exception when an incorrect database name is specified" on page 43

"Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode" error

This error occurs when either:

- The JDBC attempts to put the connection in **autocommit(true)** mode.
- A stored procedure is not created in a compatible mode.

To fix the **autocommit(true)** mode problem, let the application change the connection to chained mode using the **Connection.setAutoCommit(false)** mode, or use a **set chained on** language command.

To resolve the stored procedure problem, use the `sp_procxmode procedure_name "anymode"` command.

"JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server."

This error occurs when XA-style transactions are attempted on a server that does not have Distributed Transaction Management (DTM) installed.

To resolve this problem, use the instructions in the Sybase Manual titled: *Using Adaptive Server Distributed Transaction Management Features* to enable Distributed Transaction Management (DTM). The main steps in this procedure are:

1. Install the DTM option.
2. Check the `license.dat` file to verify that the DTM option is installed.
3. Restart the license manager.
4. Enable DTM in ISQL.
5. Restart the ASE service.

A container managed persistence (CMP) enterprise bean is causing exceptions

This error is caused by improper use of reserved words. Reserved words cannot be used as column names.

To correct this problem: Rename the variable to remove the reserved word. You can find a list of reserved words in the *Sybase Adaptive Server Enterprise Reference Manual; Volume 1: Building Blocks*, Chapter 4. This manual is available online at: <http://manuals.sybase.com/onlinebooks/group-as/asg1250e/refman>.

Sybase JDBC data source fails with "Incorrect URL format" exception in IPv6 environment

Problem

If you configure a Sybase JDBC data source through the administrative console and attempt to use it in an IPv6 environment, you can experience the following exception:

```
java.sql.SQLException: JZ0NE: Incorrect URL format
```

Cause

The Sybase JDBC drivers that are listed as selections for JDBC provider type in the administrative console do not support IPv6. The administrative console does not contain a pre-formatted template for the Sybase jConnect JDBC driver v6.0 EBF12884, which is the version required to connect to the database in an IPv6 environment.

However, you can still use the administrative console to define the Sybase jConnect JDBC driver v6.0 EBF12884.

Solution

Complete the following steps to define the Sybase jConnect JDBC driver v6.0 EBF12884:

1. In the administrative console, go to **Resources > JDBC Providers** and click **New**.
2. Select **User-defined** for the database type. This selection triggers the console to set your provider type to User-defined JDBC provider and implementation type to User-defined.
3. Click **Next** to go to the JDBC provider general configuration page.
4. In the Class path field, replace the default JAR file name with `jconn3.jar`.
5. For Implementation class name:
 - For a data source implementation that supports only one phase transactions, input `com.sybase.jdbc3.jdbc.SybConnectionPoolDataSource`
 - For an implementation that supports two phase transactions, input `com.sybase.jdbc3.jdbc.SybXADataSource`
6. Click **Apply**. Then click the **Data sources** link.
7. Click **New** to define the general properties of your data source.
8. Click **Apply** after defining the general data source properties. Then click the **Custom properties** link.
9. Define the following properties as custom properties. To set these properties, click **New**. Input the property name and a valid value for each one.
 - `databaseName`
 - `serverName`
 - `portNumber`

Executing the DatabaseMetaData.getBestRowIdentifier() method in an XA transaction causes errors

Executing the DatabaseMetaData.getBestRowIdentifier() method while in an XA transaction causes the following errors:

```
SQL Exception: The 'CREATE TABLE' command is not allowed within a multi-statement transaction in the 'tempdb' database. Calling DatabaseMetaData.getBestRowIdentifier()
```

Currently, this method fails when using Sybase. This problem occurs with other methods as well, including:

- `getBestRowIdentifier();`
- `getVersionColumns();`

- `getTablePrivileges();`
- `getProcedureColumns();`
- `getPrimaryKeys();`
- `getIndexInfo();`
- `getImportedKeys();`
- `getExportedKeys();`
- `getCrossReference();`
- `getColumns();`
- `getColumnPrivileges();`

Case 10880427 has been opened with Sybase to resolve this problem.

Sybase requirements for using the escapes and DatabaseMetaData methods

To use the escapes and DatabaseMetaData methods, you must install stored procedures on the Adaptive Server Enterprise or Adaptive Server Anywhere database where you want to use these methods. These stored procedures are also required by some of the connection methods.

To check for the presence of LOCATE ():

1. Open a Sybase **isql** command prompt.
2. Type the command **use master**.
3. Type the command **go**.
4. Type the SQL command and select * from jdbc_function_escapes.
5. Type the command **go**.

The following appears:

```
escape_name      map_string
-----
abs              abs(%1)
acos            acos(%1)
asin            asin(%1)
atan            atan(%1)
atan2           atn2(%1, %2)
ceiling         ceiling(%1)
::::::::::::::::::::::::::::::::::::::::::::::::::
locate charindex ((convert (varchar, %1)), (convert (varchar, %2)))
```

If the function does not exist, upgrade jConnect to at least Version 5.2 EBF 10635 and run the following command:

```
java IsqlApp -U sa -P -S jdbc:sybase:Tds:hostname:4100 -I %JDBC_HOME%\sp\sql_server12.sql -c go
```

Database deadlocks and XA_PROTO errors occur when using Sybase

When using Sybase with the IBM WebSphere Application Server, do one of the following to prevent database deadlocks and errors:

- Change the transaction isolation level on the connection to TRANSACTION_READ_COMMITTED. Set the isolation level on the connection for unshareable connections or, for shareable connections, define the isolation levels in the resource reference for your data source using an assembly tool.
- Modify Sybase by doing one of the following:
 - If you want to use the existing tables, modify the table locking scheme using the **alter table table name lock datarows** command to get a row lock level granularity.
 - If you want to set the system-wide locking scheme to data rows, all subsequently created tables inherit that value and have a locking scheme of data rows.

Note: You must drop your original databases and tables.

Executing a stored procedure containing a SELECT INTO command causes exception

An attempt to execute a stored procedure containing a **SELECT INTO** command results in the following exception:

```
SVR-ERROR: SQL Exception SELECT INTO command not allowed within multi-statement transaction
```

Case 10868947 has been opened with Sybase to resolve this problem.

Error is incorrectly reported about IMAGE to VARBINARY conversion

The following error is incorrectly reported:

```
com.sybase.jdbc2.jdbc.SybSQLException: Implicit conversion from data type 'IMAGE' to 'VARBINARY' is not allowed.
Use the CONVERT function to run this query.
```

The error is about a VARBINARY column only and causes confusion if you also have an IMAGE column.

Do one of the following to work around this problem:

- Use a PreparedStatement.setBytes() method instead of a PreparedStatement.setBinaryStream() method
- Use a LONG VARBINARY for the column type if you want to continue using the setBinaryStream() method. You might want to make this change because the size limit for VARBINARY is 255 bytes.

For example:

```
// *****CORRECTION*****
// setBinaryStream fails for column type of VARBINARY , use setBytes() instead
//stmt4.setBinaryStream(8,new java.io.ByteArrayInputStream(tempbyteArray),tempbyteArray.length);
stmt4.setBytes(8,tempbyteArray);
```

JDBC 1.0 standard methods are not implemented and generate a SQL exception when used

The following JDBC 1.0 standard methods are not implemented and generate a SQL exception when used:

- ResultSetMetaData.getSchemaName()
- ResultSetMetaData.getTableName() (implemented only for text and image datatypes)
- ResultSetMetaData.getCatalogName()

Sybase transaction manager fails after trying to alleviate a deadlock error

If an application encounters a deadlock, Sybase detects the deadlock and throws an exception. Because of this detection, the transaction manager calls an xa_end with a TMFAIL in it.

The call succeeds, but causes another Sybase exception, XAERR_PROTO. This exception only appears in the error log and does not cause any functional problems. All applications should continue to run, therefore no workaround is necessary.

Case 10869169 has been opened with Sybase to resolve this problem.

Starting an XA transaction when the autoCommit value of the connection is false causes error

The exception thrown is javax.transaction.xa.XAException with stack trace similar to the following:

```
at com.sybase.jdbc2.jdbc.SybXAResource.sendRPC(SybXAResource.java:711)
at com.sybase.jdbc2.jdbc.SybXAResource.sendRPC(SybXAResource.java:602)
at com.sybase.jdbc2.jdbc.SybXAResource.start(SybXAResource.java:312)
```

This problem affects you when you do both local and global transactions. If, in a local transaction, the autoCommit default value is set to *false*, and a global or XA transaction starts (either a user transaction started by you, or a container transaction started by a container), the exception occurs.

This problem is a Sybase bug as the start() method can fail unexpectedly, regardless of the value of autoCommit. Currently, there is no workaround for this problem, therefore it is not recommended that you mix local and global transactions. Case 10880792 has been opened to resolve this problem.

Sybase does not throw an exception when an incorrect database name is specified

Verify that your database name is correctly entered on the data source properties.

Most databases (DB2, Oracle, Informix®, MS SQL Server and Cloudscape) throw an exception when the database specified does not exist. But Sybase does not throw an exception when an incorrect database name is specified. Sybase generates an SQL warning and then connects to the default database. If you misspell the requested database name, Sybase connects you to the master or the default database where the table you requested is not found.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes). If you do not find your problem listed there, contact IBM Support.

JDBC trace configuration

If your application displays JDBC-related exception messages, activate the JDBC trace service. The resulting log text can help you identify the problem.

Trace strings for JDBC data sources

Turn on tracing for most database JDBC implementations through the administrative console; see the article Enabling trace, for instructions.

This method activates JDBC trace for all applications that run in the server you specify. Identify your database type by selecting the trace group WAS.database and typing one of the following trace strings in the console:

- **com.ibm.ws.database.logwriter** Trace string for databases that use the GenericDataStoreHelper. You can also use this trace string for unsupported databases.
- **com.ibm.ws.db2.logwriter** Trace string for DB2 databases.
- **com.ibm.ws.oracle.logwriter** Trace string for Oracle databases.
- **com.ibm.ws.derby.logwriter** Trace string for Derby databases.
- **com.ibm.ws.informix.logwriter** Trace string for Informix databases.
- **com.ibm.ws.sqlserver.logwriter** Trace string for Microsoft SQL Server databases.
- **com.ibm.ws.sybase.logwriter** Trace string for Sybase databases.

A few JDBC drivers require that you set trace differently, at the data source level. These drivers include:

- Microsoft SQL Server JDBC driver
- DataDirect Connect for JDBC driver for MS SQL Server

Configuring trace for these drivers through the WAS.database group results in corrupt trace information. The application server sets trace for the group at the server level, causing the trace service to begin only after your application establishes an initial connection. Because that first connection does not carry trace information, re-use of it is never tracked. Consequently the application cannot accurately match trace information to connection use.

Set trace for the previously mentioned JDBC drivers through data source custom properties. For example, use the spyAttributes custom property to enable JDBC trace for the DataDirect Connect for JDBC driver.

Consult your driver documentation for details on the custom property that enables trace for your JDBC implementation.

Additional resources

If the JDBC tracing service cannot help you isolate and fix your problem, consult the IBM Support website for WebSphere Application Server. Use the site search function to find current information on known problems and their resolutions. Locating the right troubleshooting tip can save time that you might otherwise spend on opening and tracking a PMR.

Chapter 7. Troubleshooting Dynamic caching

This page provides a starting point for finding information about the dynamic cache service, which improves performance by caching the output of servlets, commands, web services, and JavaServer Pages (JSP) files.

Dynamic caching features include replication of cache entries, cache disk offload, Edge-Side Include caching, web services, and external caching. Use external caching to control caches outside of the application server.

Troubleshooting 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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Procedure

1. Review the logs. Messages that are prefaced with *DYNA* result from dynamic cache service operations.

On the z/OS platform, check the job logs for the controller and servant.

- 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 the `DynaCacheEsi.ear` file 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 <i>Developing and deploying applications</i> 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 <i>Administering applications and their environment</i> 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: <ul style="list-style-type: none">• 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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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 you stop the server and delete all the files under the offload location.
3. If you use the WebSphere Application Server, 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**Description**

The flush attribute is set to false on the `<jsp: include>` tag in the JSP file. 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.

File system I/O activity when the application server is idle

Symptom

You experience file system I/O activity when the application server is idle. WebSphere Application Server is polling the Hierarchical File System (HFS) for the `cachespec.xml` file, even when there are no cache policies defined.

Explanation

Dynamic cache queries the file system every 30 seconds to check for updates to the cache policy file. By checking for cache policy file updates, dynamic cache can automatically update any changed cache policies.

Possible response

The dynamic cache is enabled by default because several system components rely on it for performance reasons. You can disable the dynamic cache service by using a `wsadmin` script or through the administrative console. See the *Administering applications and their environment* PDF for more information.

CAUTION:

By disabling dynamic cache you might experience slower performance of security because it uses the distributed map functionality of the dynamic cache service.

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.

Restriction: 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(); %>
```

Service integration bus messages are repeated in the logs of the cluster members hosting a production application

Problem

In a multi-cell environment, the following messages are repeated in the logs of the cluster members hosting the production application:

```
[time_stamp] CWSIT0007W: It is not
possible to contact the bootstrap server at
9.9.9.9:7299:BootstrapBasicMessaging because of exception:
com.ibm.websphere.sib.exception.SIResourceException: CWSIC1001E:
A client attempted to connect with a remote messaging engine (9.9.9.9:7299 -
BootstrapBasicMessaging) but the connection cannot be completed. Ensure the
messaging engine is started:
exception com.ibm.ws.sib.jfapchannel.JFapConnectFailedException: CWSIJ0063E: A
network connection to host name 9.9.9.9, port 7299 cannot be established...
[time_stamp] 00000023 SystemOut 0 RemoteInvalidator unable to connect to ...
```

Recommended response

First, ensure each service integration bus member in the remote cell is started. Next, ensure that the SIB_ENDPOINT_ADDRESS port is specified correctly for each service integration bus member in the remote cell or core group. If the wrong port is specified, delete the outbound configuration "--setup=dynacacheOutSIB --delete..." and reconfigure it using the correct port. When everything is working correctly, a message similar to the following message displays in the logs:

```
[time_stamp] 0000000a SystemOut 0 RemoteInvalidator successfully connected
to DynacacheDestination-edgeaphid10Cell101-cg1 using
```

Platform messaging component cannot authenticate the user ID

Explanation

In a multi-cell environment, a service integration bus member has the following error:

```
[time_stamp] 00000022 SibMessage E [:] CWSII0050E: The Platform Messaging Component can not authenticate the user ID.
```

This message might be misleading because the dynamic cache service does not use secure buses.

Recommended response

This message might indicate one of the following problems:

The service integration bus server is receiving a request for a bus destination that does not exist. Ensure the correct remote cell name was used when executing "--setup=dynacacheOutSIB" and "--setup=dynacacheECA" in the sending cell.

No service integration bus members are available. This problem often occurs because the wrong cluster is specified when doing the inbound setup "--setup=dynacacheInSIB". Ensure the service integration bus cluster is specified for the "--cluster" option and not some other cluster.

The messaging engine's unique ID does not match that found in the data store

Explanation

In a multi-cell environment, a service integration bus member has the following error:

```
[DynacacheBus-edgeaphid10Cell101-cg2:edgeaphid10Node01.cg2SIBServer-DynacacheBus-edgeaphid10Cell101-cg2]
CWSIS1535E: The messaging engine's unique id does not match that found in the data store. ME_UUID=D520787E8CA7F18A, ME_UUID(DB)=980C0B42B3A904F3
[time_stamp] 0000002f SibMessage I
[DynacacheBus-edgeaphid10Cell101-cg2:edgeaphid10Node01.cg2SIBServer-DynacacheBus-edgeaphid10Cell101-cg2]
CWSIS1546I: The messaging engine, ME_UUID=D520787E8CA7F18A, INC_UUID=7228ea45e216f3ef, has lost an existing lock or failed to gain an initial lock on the data store.
[time_stamp] 0000002f SibMessage I
[DynacacheBus-edgeaphid10Cell101-cg2:edgeaphid10Node01.cg2SIBServer-DynacacheBus-edgeaphid10Cell101-cg2]
CWSIS1519E: Messaging engine edgeaphid10Node01.cg2SIBServer-DynacacheBus-edgeaphid10Cell101-cg2 cannot obtain the lock on its data store, which ensures it has exclusive access to the data.
[time_stamp] 0000002d SibMessage E
[DynacacheBus-edgeaphid10Cell101-cg2:edgeaphid10Node01.cg2SIBServer-DynacacheBus-edgeaphid10Cell101-cg2]
CWSIS0002E: The messaging engine encountered an exception while starting. Exception: com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E: The data source has produced an unexpected exception: com.ibm.ws.sib.msgstore.persistence.DatasourceWrapperStateException: New connections cannot be provided because the persistence layer has been stopped at com.ibm.ws.sib.msgstore.persistence.impl.PersistentMessageStoreImpl.start (PersistentMessageStoreImpl.java:188) at com.ibm.ws.sib.msgstore.impl.MessageStoreImpl.start (MessageStoreImpl.java:1175) at com.ibm.ws.sib.admin.impl.JsMessagingEngineImpl.start (JsMessagingEngineImpl.java:491)
```

Recommended response

Stop the server and delete the directory in `WAS_INSTALL_ROOT\profiles\AppSrv01\filestores\com.ibm.ws.sib`, that corresponds to the service integration bus member and restart the server.

dynacacheJMSSIB script can not locate DynacacheMessageHandler.ear file

Explanation

In a multi-cell environment, the dynacacheJMSSIB script can not locate DynacacheMessageHandler.ear file, and the following message appears in the log:

```
WASX7017E: Exception received while running file
"../../../../util/dynacacheJMSSIB.py"; exception information:
com.ibm.ws.scripting.ScriptingException: WASX7115E: Cannot read input file
"/opt/WAS/6.1/cf270928.19/profiles/AppSrv01/logs/DynacacheMessageHandler.ear"
```

Recommended response

Run the dynacacheJMSSIB script from *WAS_INSTALL_ROOT/profiles/PROFILE_NAME/bin* so that it can locate *../../../../util/dynacacheJMSSIB.py* and *../../../../installableApps/DynacacheMessageHandler.ear* at the appropriate relative paths.

Chapter 8. Troubleshooting EJB applications

This page provides a starting point for finding information about enterprise beans.

Based on the Enterprise JavaBeans (EJB) specification, enterprise beans are Java components that typically implement the business logic of Java 2 Platform, Enterprise Edition (J2EE) applications as well as access data.

Troubleshooting Enterprise JavaBeans applications

Enterprise bean and EJB container troubleshooting tips

If you are having problems starting an Enterprise JavaBeans (EJB) container, or encounter error messages or exceptions that are generated by an EJB container, follow these steps to resolve the problem:

- Use the administrative console to verify that the application server which hosts the container is running.
- Browse the logs for the application server which hosts the container. Look for the message `server_name` open for e-business in the server log files. If the message does not display, or if you see the message, problems occurred during startup, browse the server log files for details.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

If none of these steps solves the problem, check to see if the problem is identified and documented in the following topics: 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 to resolve this problem. Before opening a PMR, see the IBM Support page.

Application client log error indicates missing JAR file

The following error message appears in the client log file because a Java archive (JAR) file is missing from the classpath on the client machine. The Object Request Broker (ORB) needs this file to unmarshal the nested exception that is part of the EJB exception, returned by the server to the client application. For example, if the EJB returns a DB2® JCC SQL exception nested inside of the EJB exception that it returns to the client, the ORB is not able to unmarshal the nested exception if the `db2jcc.jar` file that contains the DB2 SQL exception is not in the client classpath.

```
java.rmi.MarshalException: CORBA MARSHAL 0x4942f89a No; nested exception is:
org.omg.CORBA.MARSHAL: Unable to read value
from underlying bridge : Custom marshaling (4) Sender's class does not match
local class vmcid: 0x4942f000 minor code: 2202 completed: No*
```

To avoid this error, include the JAR file that contains the class for the nested exception that is returned in the EJB exception.

Enterprise bean cannot be accessed from a servlet, a JSP file, a stand-alone program, or another client

Use these troubleshooting tips for problems related to accessing enterprise beans.

What error are you seeing?

- **java.lang.NoSuchMethodError** message
- **javax.naming.NameNotFoundException: Name *name* not found in context "local" message** when access is attempted
- **BeanNotReentrantException** is thrown
- **CSITransactionRolledbackException / TransactionRolledbackException** is thrown
- Call fails, Stack trace beginning **EJSContainer E Bean method threw exception [exception_name]** found in JVM log file.
- Call fails, **ObjectNotFoundException** or **ObjectNotFoundLocalException** when accessing stateful session EJB found in JVM log file.
- Attempt to start container managed persistence (CMP) Enterprise JavaBeans (EJB) module fails with **javax.naming.NameNotFoundException: *dataSourceName***
-
- **Message BBOT0003W is issued**
- Symptom: **CNTR0001W: A Stateful SessionBean could not be passivated**

If the client is remote to the enterprise bean, which means, running in a different application server or as a stand-alone client, browse the logs of the application server hosting the enterprise bean, as well as log files of the client.

java.lang.NoSuchMethodError

When trying to invoke a method on a Session Bean, a java.lang.NoSuchMethodError occurred.

The following is an example of this type of error:

When trying to invoke the method xSLTStory4Session on a bean, a java.lang.NoSuchMethodError displayed.

```
CNTR0020E: EJB threw an unexpected (non-declared) exception during invocation of method "xSLTStory4Session" on bean "BeanId
Exception data: java.lang.NoSuchMethodError: paysession/ejb3/PaySessionFacadeRemote.paySession(Ljava/lang/String;)Ljava/lang
```

In this case, there are multiple versions of the PaySessionFacadeRemote interface in multiple modules within the EAR file. One of the interfaces does not contain the expected method.

The stub class was generated by the run time from the first version of the PaySessionFacadeRemote interface in the class path of the EAR file which was the incorrect version.

ObjectNotFoundException or ObjectNotFoundLocalException when accessing stateful session EJB

A possible cause of this problem is that the stateful session bean timed out and was removed by the container. This event must be addressed in the code, according to the EJB 2.1 and later specification. You can review the EJB 2.1 and 3.0 specifications at <http://java.sun.com/products/ejb/docs.html>.

Stack trace beginning "EJSContainer E Bean method threw exception [exception_name]" found in JVM log file

If the exception name indicates an exception thrown by an IBM class that begins with "com.ibm...", then search for the exception name within the information center, and in the online help as described below. If "exception name" indicates an exception thrown by your application, contact the application developer to determine the cause.

javax.naming.NameNotFoundException: Name name not found in context "local"

A possible reason for this exception is that the enterprise bean is not local (not running in the same Java virtual machine [JVM] or application server) to the client JSP, servlet, Java application, or other enterprise bean, yet the call is to a "local" interface method of the enterprise bean. If access worked in a development environment but not when deployed to WebSphere Application Server, for example, it might be that the enterprise bean and its client were in the same JVM in development, but are in separate processes after deployment.

To resolve this problem, contact the developer of the enterprise bean and determine whether the client call is to a method in the local interface for the enterprise bean. If so, have the client code changed to call a remote interface method, or to promote the local method into the remote interface.

References to enterprise beans with local interfaces are bound in a name space local to the server process with the URL scheme of `local:`.

BeanNotReentrantException is thrown

This problem can occur because client code, typically a servlet or JSP file, is attempting to call the same stateful SessionBean from two different client threads. This situation often results when an application stores the reference to the stateful session bean in a static variable, uses a global (static) JSP variable to refer to the stateful SessionBean reference, or stores the stateful SessionBean reference in the HTTP session object. The application then has the client browser issue a new request to the servlet or JSP file before the previous request has completed.

To resolve this problem, ask the developer of the client code to review the code for these conditions.

CSITransactionRolledbackException / TransactionRolledbackException is thrown

An enterprise bean container creates these high-level exceptions to indicate that an enterprise bean call did not complete. When this exception is thrown, browse the logs to determine the underlying cause.

Some possible causes include:

- The enterprise bean might throw an exception that was not declared as part of its method signature. The container is required to roll back the transaction in this case. Common causes of this situation are where the enterprise bean or code that it calls creates a `NullPointerException`, `ArrayIndexOutOfBoundsException`, or other Java runtime exception, or where a BMP bean encounters a JDBC error. The resolution is to investigate the enterprise bean code and resolve the underlying exception, or to add the exception to the problem method signature.
- A transaction might attempt to do additional work after being placed in a "Marked Rollback", "RollingBack", or "RolledBack" state. Transactions cannot continue to do work after they are set to one of these states. This situation occurs because the transaction has timed out which, often occurs because of a database deadlock. Work with the application database management tools or administrator to determine whether database transactions called by the enterprise bean are timing out.
- A transaction might fail on commit due to dangling work from local transactions. The local transaction encounters some "dangling work" during commit. When a local transaction encounters an "unresolved action" the default action is to "rollback". You can adjust this action to "commit" in an assembly tool. See the assembly tool information center on how to adjust

Attempt to start EJB module fails with "javax.naming.NameNotFoundException dataSourceName_CMP"exception

This problem can occur because:

- When the DataSource resource was configured, container managed persistence was not selected.

- To confirm this problem, in the administrative console, browse the properties of the data source given in the NameNotFoundException. On the Configuration panel, look for a check box labeled **Container Managed Persistence**.
- To correct this problem, select the check box for **Container Managed Persistence**.
- If container managed persistence is selected, it is possible that the CMP DataSource was not bound into the namespace.
 - Look for additional naming warnings or errors in the status bar, and in the hosting application server logs. Check any further naming-exception problems that you find by looking at the topic “Application access problems” on page 82.

Message BBOT0003W is issued

Message BBOT0003W indicates a transaction timeout. The timeout might result in the abnormal termination of the servant where the transaction is running.

- The default timeout value for enterprise bean transactions is 120 seconds. After this time, the transaction times out and the connection closes.
- If the transaction legitimately takes longer than the specified timeout period, on the administrative console:
 1. Go to **Manage Application Servers > server_name**
 2. Select the **Transaction Service properties** page
 3. Increase the **Total transaction lifetime timeout** value
 4. **Save** the configuration

Note: z/OS will use the value you set for **Total transaction lifetime timeout** as the default transaction timeout setting. If you set a value for this property that is greater than the maximum transaction timeout value, z/OS will use the maximum transaction timeout value as the default.

Symptom:CNTR0001W: A Stateful SessionBean could not be passivated

This error can occur when a Connection object used in the bean is not closed or nulled out.

To confirm this is the problem, look for an exception stack in the logs for the EJB container that hosts the enterprise bean, and looks similar to:

```
StatefulPassi W CNTR0001W:
A Stateful SessionBean could not be passivated: StatefulBean0
(BeanId(XXX#YYY.jar#ZZZ),
state = PASSIVATING)
java.io.NotSerializableException: com.ibm.ws.rsadapter.jdbc.WSJdbcConnection
at java.io.ObjectOutputStream.writeObject((Compiled Code))
at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
at java.io.ObjectOutputStream.outputClassFields((Compiled Code))
at java.io.ObjectOutputStream.defaultWriteObject((Compiled Code))
at java.io.ObjectOutputStream.writeObject((Compiled Code))
at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
at com.ibm.ejs.container.passivator.StatefulPassivator.passivate((Compiled Code))

at com.ibm.ejs.container.StatefulBean0.passivate((Compiled Code))
at com.ibm.ejs.container.activator.StatefulASActivationStrategy.atUnitOfWorkEnd
((Compiled Code))
at com.ibm.ejs.container.activator.Activator.unitOfWorkEnd((Compiled Code))
at com.ibm.ejs.container.ContainerAS.afterCompletion((Compiled Code))
```

where *XXX,YYY,ZZZ* is the bean name.

To correct this problem, the application must close all connections and set the reference to null for all connections. Typically this activity is done in the `ejbPassivate()` method of the bean. Also, note that the bean must have code to reacquire these connections when the bean is reactivated. Otherwise, there are NullPointerExceptions when the application tries to reuse the connections.

Troubleshooting access intents for EJB 2.x entity beans

Access intent exceptions

The following are exceptions that occur in response to the application of access intent policies.

com.ibm.ws.ejbpersistence.utilpm.OptimisticUpdateFailedException

If an update fails under optimistic concurrency because fields changed within another transaction between load and store requests, an `OptimisticUpdateFailedException` is raised and the commit fails.

com.ibm.ws.ejbpersistence.utilpm.PersistenceManagerException

If the method that drives the `ejbLoad()` method is configured to be read-only but updates are then made within the transaction that loaded the bean's state, an exception is thrown during invocation of the `ejbStore()` method, and the transaction is rolled back. Likewise, the `ejbRemove()` method cannot succeed in a transaction that is set as read-only. If an update hint is applied to methods of entity beans with bean-managed persistence, the same behavior and exception results. The forwarded exception object contains the message string `PMGR1103E: update instance level read only bean beanName`

This exception is also thrown if the applied access intent policy cannot be honored because a finder, `ejbSelect`, or container-managed relationship (CMR) accessor method returns an inherently read-only result. The forwarded exception object contains the message string `PMGR1001: No such DataAccessSpec - methodName`

The most common occurrence of this error is when a custom finder that contains a read-only EJB Query Language (EJB QL) statement is called with an applied access intent of `wsPessimisticUpdate` or `wsPessimisticUpdate-Exclusive`. These policies require the use of a `USE AND KEEP UPDATE LOCKS` clause on the SQL `SELECT` statement to be executed, but a read-only query cannot support `USE AND KEEP UPDATE LOCKS`. Other examples of read-only queries include joins; the use of `ORDER BY`, `GROUP BY`, and `DISTINCT` keywords.

To eliminate the exception, edit the EJB query so that it does not return an inherently read-only result or change the access intent policy being applied.

- If an update access is required, change the applied access intent setting to `wsPessimisticUpdate-WeakestLockAtLoad` or `wsOptimisticUpdate`.
- If update access is not truly required, use `wsPessimisticRead` or `wsOptimisticRead`.
- If connection sharing between entity beans is required, use `wsPessimisticUpdate-WeakestLockAtLoad` or `wsPessimisticRead`.

com.ibm.websphere.ejb.container.CollectionCannotBeFurtherAccessed

If a lazy collection is driven after it is no longer in scope, and beyond what has already been locally buffered, a `CollectionCannotBeFurtherAccessed` exception is thrown.

com.ibm.ws.exception.RuntimeWarning

If an application is configured incorrectly, a run-time warning exception is thrown as the application starts; startup is ended. You can validate an application's configuration by choosing the `verify` function. Some examples of misconfiguration include:

- A method configured with two different access intent policies
- A method configured with an undefined access intent policy

Access intent troubleshooting tips

The following frequently asked questions involving access intent are answered.

Oracle database fails when no access policies are applied

No access intent policies have been applied and the application runs with a DB2 database, but it fails with an Oracle database with the following message:

```
com.ibm.ws.ejbpersistence.utilpm.PersistenceManagerException: PMGR1001E: No such DataAccessSpec :FindAllCustomers. The backend datastore does not support the SQLStatement needed by this
```

AccessIntent: (pessimistic update-weakestLockAtLoad)(collections: transaction/25) (resource manager prefetch: 0) (AccessIntentImpl@d23690a).

If you have not configured access intent, all of your data is accessed under the default access intent policy (wsPessimisticUpdate-WeakestLockAtLoad). On DB2 the weakest lock is share. On Oracle databases, however, the weakest lock is update; this means that the SQL query must contain a FOR UPDATE clause. To avoid this problem, try to apply an access intent policy that supports optimistic concurrency.

Calling a finder method displays an InconsistentAccessIntentException at run time

This can occur when you use method-level access intent policies to apply more control over how a bean instance is loaded. This exception indicates that the entity bean was previously loaded in the same transaction. This could happen if you called a multifinder method that returned the bean instance with access intent policy X applied; you are now trying to load the second bean again by calling its findByPrimaryKey method with access intent Y applied. Both methods must have the same access intent policy applied.

Likewise, if the entity was loaded once in the transaction using an access intent policy configured on a finder, you might have called a container-managed relationship (CMR) accessor method that returned the entity bean configured to load using that entity's default access intent.

To avoid this problem, ensure that your code does not load the same bean instance twice within the same transaction with different access intent policies applied. Avoid the use of method-level access intent unless absolutely necessary.

An InconsistentAccessIntentException displays in a container-managed relationship with two beans

There are two beans in a container-managed relationship. The findByPrimaryKey method is called on the first bean and then the getBean2 method is called and a CMR accessor method is called, on the returned instance and an InconsistentAccessIntentException displays.

You are probably using read-ahead. When you loaded the first bean, you caused the second bean to be loaded under the access intent policy applied to the finder method for the first bean. However, you have configured your CMR accessor method from the first bean to the second with a different access intent policy. CMR accessor methods are really finder methods in disguise; the run-time environment behaves as if you were trying to change the access intent for an instance you have already read from persistent store.

To avoid this problem, beans configured in a read-ahead hint are all driven to load with the same access intent policy as the bean to which the read-ahead hint is applied.

A bean with a one-to-many relationship to a second bean displays an UpdateCannotProceedWithIntegrityException error when an instance of the second bean is added to the first bean's collection

A bean with a one-to-many relationship to a second bean displays an UpdateCannotProceedWithIntegrityException error when an instance of the second bean is added to the first bean's collection. The first bean has a pessimistic-update intent policy applied.

The second bean probably has a read intent policy applied. When you add the second bean to the first bean's collection, you are not updating the first bean's state, you are implicitly modifying the second bean's state. The second bean contains a foreign key to the first bean, which is modified.

To avoid this problem, ensure that both ends of the relationship have an update intent policy applied if you expect to change the relationship at run time.

Troubleshooting JPA applications

Use this information to find various known problems with Java Persistence API (JPA) applications.

Procedure

- Review messages that are related to transactions.

Under certain conditions, a message like the following might be logged: Unable to execute {0} on a WebSphere managed transaction. WebSphere does not support direct manipulation of managed transactions.

This error is probably caused by a data source that is not configured correctly as <non-jta-data-source>. See the information center topic, *Associating persistence providers and data sources*, on how to configure a data source to be used as a <non-jta-data-source>.

- Troubleshoot classes that have not been enhanced by run time.

It is difficult to diagnose these situations. Sometimes you can trace the problem back to a lack of OpenJPA enhancement of entity classes. Examples of these situations might be detecting when entities are not *persistence capable*. Look for a message like the following in the log: This configuration disallows runtime optimization, but the following listed types were not enhanced at build time or at class load time with a Java agent: "{0}".

This message indicates that the expected runtime enhancement did not take place on the listed entity types. In most cases, this error is just a build time failure and the PCEnhancer must be run on the listed classes. It can also indicate a more involved problem, especially if the container class loader transform is expected to do the entity enhancement.

- Troubleshoot issues with closed cursors.

The following is a DB2 exception located in the log
org.apache.openjpa.persistence.PersistenceException:

```
[ibm][db2][jcc][10120][10898] Invalid operation: result set is closed can be a  
WebSphere Application Server configuration problem.
```

By default, the application server configures the `resultSetHoldability` custom property with a value of 2 (`CLOSE_CURSORS_AT_COMMIT`). This property causes DB2 to close its `resultSet/cursor` at transaction boundaries. Despite the DB2 default `resultSetHoldability` value of 1 (`HOLD_CURSORS_OVER_COMMIT`), the application server retains the default value of 2 to avoid breaking compatibilities with previous releases of the application server. You can change the default.

Attention: If this data source is an XA data source, define a new custom property on the data source where `property_name = downgradeHoldCursorsUnderXa` and `boolean value = true`.

Note: In Version 8.0, if a DB2 XA data source is used, the following configuration are required to overcome the *result set closed* problem:

1. Use DB2 Version 9.1 FP4 (for APAR IZ18072) or a later version.
2. Add custom property `name="downgradeHoldCursorsUnderXa", value="true"` and `type="java.lang.Boolean"`
3. Add custom property `name="resultSetHoldability", value="1"`, and `type="java.lang.Integer"`

- Avoid multi-threaded EntityManager usage. If an exception occurs with the following message text, you might be accidentally supporting the usage of an EntityManager across multiple threads.

Per the JPA specification, EntityManagers are meant to be used by a single thread. You might receive an exception message that is like the following (in this context, brokers and EntityManagers are essentially the same thing):

```
Multiple concurrent threads attempted to access a single broker. By default brokers are not thread safe;  
if you require and intend a broker to be accessed by more than one thread, set the openjpa.Multithreaded property  
to true to override the default behavior.
```

There are some ways to address this issue:

–

Note: Use the get-use-close pattern when using application-managed (extended) persistence contexts. Remember to close the EntityManager before leaving the method that used the EntityManager. Leaving the EntityManager open can accidentally cause other threads to use the same EntityManager instance at the same time and might consume system resources.

- Change the application to use container-managed persistence contexts by injecting the EntityManager instance through the @PersistenceContext annotation, if the programming model for the application is amenable to this change. Essentially, this enforces the get-use-close pattern by supporting the container to do the management.
- As documented in this exception text, a quick workaround is to configure OpenJPA to require multi-threaded access to the EntityManagers in the openjpa.Multithreaded property. Enabling this property can introduce unnecessary overhead.

Note: Instance variables for servlets are automatically shared by all instances of the servlet. Using the @PersistenceContext annotation on a servlet instance variable can unintentionally result in multi-threaded access to the same EntityManager. In addition, any EntityManagers injected in this manner are not cleaned up by the container until the application is stopped. For servlets, it is a better choice to use the @PersistenceUnit annotation to inject an EntityManagerFactory.

- If you are using optimistic locking, be aware of version conditions. In the JPA architecture, the persistence provider uses the version property to perform optimistic locking and concurrency semantics for a persisting entity; for example:

```
@Entity
public class VersionedTimestampEntity
{
    @Id    private int id;
    @Version private java.sql.Timestamp version;
    ....
}
```

The provider updates the version property to a new value when an entity is written to the database. During an entity merge operation, the persistence provider examines this versioned property to determine if the entity that is being merged is a stale copy of the entity. If the operation failed due to the stale version condition, an OptimisticLockException occurs. The version property can be one of these types:

- int
- Integer
- short
- Short
- long
- Long
- Timestamp.

Note: If you use the Timestamp type for the version property, the application must be aware of behavior that can be exhibited due from the implementation precision that is used for creating the Timestamp object. The typical implementation uses the System.currentTimeMillis method. The time precision of this method is platform-specific. For example, the precision is 15 ms for 32-bit Windows and 2 ms for z/OS.

If an entity is persisted, and the entity is then fetched and updated in a separate persistence context that is within the precision window of the platform, the persistence provider cannot detect the optimistic locking and concurrent condition. As a result, an OptimisticLockException might not be thrown and data integrity is compromised.

- Troubleshoot database constraint violations when working with entities.

The JPA providers included with WebSphere Application Server use a constraint update manager to determine the order of SQL requests to the database based on each entity configuration. This update manager can rearrange the order of SQL requests to the database. This alleviates an application from knowing the order of entity manager operations required to perform its business logic and optimizes database performance using underlying batching support.

Since the JPA provider does not assume that there are implied database constraints for relationships between entities, if there are constraints in the database, for example, a foreign key constraint, the JPA provider might not issue SQL statements in the wanted order. Under these conditions, the following or similar exception might occur:

```
com.ibm.db2.jcc.b.SqlException: DB2 SQL error: SQLCODE: -530, SQLSTATE: 23503, SQLERRMC: xxxxxx
```

There are some ways to address this issue:

- The JPA provider can be configured to read constraint information from the database and apply to the update manager at run time by adding the following configuration property to the persistence unit.

```
<property name="openjpa.jdbc.SchemaFactory" value="native(ForeignKeys=true)" />
```

- The application can enhanced entity to use `@ForeignKey` annotation to indicate to the JPA provider which relationships have foreign key constraints.

```
public class Person {
    @ManyToOne
    @ForeignKey
    private Address address;
    ....
}
```

- The application can take on the responsibility of ordering the SQL statements by adding the following configuration property to the persistence unit.

```
<property name="openjpa.jdbc.UpdateManager" value="operation-order" />
```

With this configuration option present, the JPA provider starts the SQL statements in the same order as the entity operations were requested. The application must be aware of any interdependencies between entities.

- Remove the constraints from the database.

- Troubleshoot using the OpenJPA MappingTool ANT task.

The MappingTool ANT task provided by OpenJPA uses a temporary class loader to load the JDBC driver classes. This temporary class loader might have trouble loading some JDBC drivers such as DB2.

When you run the MappingTool ANT task, you see an error similar to the following:[mappingtool]

```
java.lang.UnsatisfiedLinkError: com/ibm/jvm/Trace.initTrace([Ljava/lang/String;[Ljava/lang/String;)V [mappingtool] at com.ibm.jvm.Trace.initializeTrace(Trace.java:94) [mappingtool] at com.ibm.jvm.Trace.<clinit>(Trace.java:59) [mappingtool] at java.lang.J9VMInternals.initializeImpl(Native Method) [mappingtool] at java.lang.J9VMInternals.initialize(J9VMInternals.java:200) [mappingtool] at java.lang.Class.forNameImpl(Native Method) [mappingtool] at java.lang.Class.forName(Class.java:136) [mappingtool] at com.ibm.db2.jcc.a.o.n(o.java:577) [mappingtool] at com.ibm.db2.jcc.a.o.<clinit>(o.java:329)
```

In order to use the mapping tool, you can disable the temporary class loader by adding the `tmpClassLoader=false` argument to the ANT task. Two example ANT scripts follow:

This example exhibits the problem:

```
<taskdef name="mapping" classname="org.apache.openjpa.jdbc.ant.MappingToolTask" classpathref="jpa.cp"/>
.
.
.
<target name="map.broken">
  <mapping>
    <!-- this exhibits the problem -->
    .
    .
    .
  </mapping>
</target>
```

This example prevents the problem:

```
<taskdef name="mapping" classname="org.apache.openjpa.jdbc.ant.MappingToolTask" classpathref="jpa.cp"/>
. . .
<target name="map.fixed">
  <mapping tmpClassLoader="false">
    <!-- this will work -->
    . . .
  </mapping>
</target>
```

- Impact of DataCache on inconsistent domain models

When an application persists an inconsistent domain model and later retrieves the entities in a separate persistence context, the retrieved entities are different depending on if the DataCache is active or not.

For example, consider bidirectional, one-to-one relationship between two entities mapped by a single foreign key column in the database. It is possible for an application to set the relation fields in the entities inconsistently. When such inconsistent values are mapped to the database, the database records become consistent only because a single foreign key column expresses the bidirectional relationships. DataCache is active, however, then DataCache captures the inconsistent in-memory entity states. Therefore, when an application persists a pair of entities related in a bidirectional relation but their relation fields are set to inconsistent values and later retrieves the entities in a different persistence context, the bidirectional relationship either remains inconsistent or becomes consistent depending on if the DataCache is used.

When multiple fields are mapped to the same column but set to different values is another example where an inconsistent entity state is persisted but retrieved as consistent in a separate persistence context. In this case, the introduction of DataCache also causes an entity realized in a separate persistence context to retain different and inconsistent values while without a DataCache, an entity holds the same value for both the fields.

bprac: Avoid populating the application domain model inconsistently. It is also possible to configure `openjpa.InverseManager` property to detect certain inconsistencies such as bidirectional relationship.

- Troubleshoot problems with the MappingTool and `@ForeignKey` annotation with Sybase.

The OpenJPA mapping tool is unable to create ForeignKey constraints when used with Sybase Adaptive Server. As a result, the foreign key constraints must be created manually.

- Troubleshoot the DB2 Optim™ pureQuery Runtime configuration options.

If you are using DB2 Optim pureQuery Runtime with the WebSphere Application Server JPA provider, you must set the following property in the `persistence.xml` file:

```
<property name="openjpa.Compatibility" value="StrictIdentityValues=true"/>
```

If you must set a different compatibility option, for example `ReloadOnDetach=false`, you must specify both options as parameters of the same property statement in the `persistence.xml` file. For example :

```
<property name="openjpa.Compatibility" value="StrictIdentityValues=true,ReloadOnDetach=false"/>
```

Logging applications with JPA

Logging supports viewing, tracing, and troubleshooting the runtime behavior of an application. Java Persistence API (JPA) provides a flexible logging system that is integrated with the application server to assist you in troubleshooting problems.

About this task

Logging channels

Attention: Logging can have a negative impact on performance. Limit or disable logging when you run any performance tests.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

In addition to the channels used by OpenJPA, a trace group named `openjpa` enables channels that are prefixed with “`openjpa.`” Specifying “`openjpa`” for a trace group overrides any other trace group specification that is specific to a channel; for example:

```
openjpa.Runtime=debug:openjpa.jdbc.SQL=all
```

```
openjpa=all
```

Note: The `openjpa.Log` property is ignored if it is defined in a container-managed persistence unit that uses the persistence providers that are provided with the application server. In this case, you must use the standard trace specification for the application server.

OpenJPA and JPA for WebSphere Application Server implement logging channels to which message data, trace data, and debugging data can be recorded to a configurable repository. The JPA component creates the logging channel at run time and assigns a channel name for identification. The component writes information to the configured repository through the channel. OpenJPA and JPA for WebSphere Application Server create the following channels:

- **openjpa.Tools** - Command line and Ant server tools
- **openjpa.MetaData** - Metadata information
- **openjpa.Enhance** - Enhancement and runtime class generation
- **openjpa.Runtime** - Messages generated during run time
- **openjpa.Query** - Query information
- **openjpa.DataCache** - L2 data cache plug-in information
- **openjpa.jdbc.JDBC** - JDBC connection information
- **openjpa.jdbc.SQL** - Detailed SQL execution statements and information
- **openjpa.jdbc.SQLDiag** - Additional SQL diagnostic information for the entity operations create, retrieve, update, and delete.
- **openjpa.jdbc.Schema** - Details about operations on the database schema
- **wsjpa.pdq** - Trace of all interactions between store manager and PDQ run time
- **wsjpa.Sqlgen** - Diagnostic trace for `wsdbgen` program

Logging levels

Each of the logging channels use logging levels to control which messages are recorded. The following logging levels are supported by the JPA architecture:

- **TRACE** - the most detailed option
- **INFO** - information related to the specific channel
- **WARN** - warning messages
- **ERROR** - error condition messages
- **FATAL** - fatal condition messages

By using a particular logging channel together with logging levels, you can control the types of logging messages and the amount of logging messages that are recorded.

Note: These logging functions apply only to OpenJPA and JPA for the application server. Logging functions that are provided in implementations of a third-party persistence provider are not covered. However, if the logging output from a third-party persistence provider is directed to the Java System.out or System.err file output streams, the messages are handled by the environment accordingly at run time.

Logging in the application server

The default JPA persistence provider that is supported by the application server records messages and tracing data that are automatically integrated into the RAS component. Alternatively, OpenJPA implements a custom logger to route messages from OpenJPA channels to the channels of the application server.

The channel names that are supported by OpenJPA are used as the trace group names in the trace level for the application server. The mappings of OpenJPA logging levels to trace levels in the application server are:

Table 4. Mapping OpenJPA logging levels to application server trace levels. The mappings of OpenJPA logging levels to trace levels in the application server are:

OpenJPA logging level	Trace level for the application server
TRACE	debug
INFO	info
WARN	warning
ERROR	error
FATAL	fatal

Logging in a client container and standalone Java application

OpenJPA logging uses the basic logging framework and output format:

```
millis [persistence-unit name]level[thread identifier] channel - message
```

Important: When using IBM Optim PureQuery Run time, the PDQ store manager also uses JDBC in some situations, such as for large result set processing. When tracing all calls to the database, you must trace both JDBC and PDQ.

Example:

```
property name="wsjpa.Log" value="SQL=TRACE"/
```

This traces the SQL and input parameter values.

```
property name="wsjpa.Log" value="pdq=TRACE, JDBC=TRACE"/
```

This performs a detailed trace of calls to the IBM Optim PureQuery Runtime and any calls to JDBC. If you are using pureQuery and must trace calls to the database, you must perform both traces.

The default logging system accepts the following parameters:

- **File:** The name of the file to which the application server logs information. You can also use standard output stdout or standard error stderr to send messages. By default, JPA sends log messages to standard error.
- **DefaultLevel:** The default logging level for channels that are not configured. The values can be TRACE, INFO, WARN, and ERROR. The default setting is INFO.

- **DiagnosticContext:** A string that is placed at the beginning of all log messages. If a DiagnosticContext is not supplied and an openjpa.id property is available, the openjpa.id value is used.
- **<channel>:** The channel that is being logged. This parameter can be used to configure the logging level of the channel.

Procedure

1. Open the persistence.xml file for the application that you want to modify.
2. Add a property name tag to the XML schema named openjpa.log. For example:

```
<property name="openjpa.log" .../>
```
3. Add additional parameters. For example:

```
<property name="openjpa.log" value="DefaultLevel=WARN .../>
```

Note: To reduce overhead by disabling logging, set the openjpa.log property to NONE and proceed to Step 5.

4. Designate the logging channels and the logging level. For example:

```
<property name="openjpa.log" value="DefaultLevel=WARN, Runtime=INFO, Tool=INFO" .../>
```
5. Save changes to the file.

Results

The next time the application is started the JPA component logs all channels at the WARN logging level and the Runtime and Tool channels at the INFO level.

What to do next

OpenJPA supports the substitution of other logging methods. Read the logging section of the Apache OpenJPA User Guide for more information and examples.

Enabling enhanced tracing for JPA

In some situations the trace information generated by the Java Persistence API (JPA) providers shipped with WebSphere Application Server might not be adequate to diagnose a problem. In these situations, an extended trace mechanism can be enabled to generate additional information in the trace file. Extended trace can function only with IBM-shipped persistence providers. It does not work with third-party providers, including alternate versions of OpenJPA bundled within an application or configured as a shared library.

About this task

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Enabling enhanced tracing in a WebSphere Application Server environment

Enhanced JPA tracing for an application running on WebSphere Application Server can be enabled with a few simple steps using wsadmin scripting or the administration console. The steps in this topic describe how to configure enhanced tracing using the administration console. This process changes server settings, so it is good practice to back up your server configuration before proceeding.

Procedure

1. Enable the trace agent. A trace agent must be enabled per application server by passing an argument to the server Java Virtual Machine (JVM). The trace agent can be enabled using the administrative console by following these steps:
 - a. In the navigation pane, select **Servers**. Select **Application Servers**.
 - b. In the server list pane, select the server that needs the enhanced JPA trace. If multiple servers provide JPA functionality to your application, these steps must be followed for every server.
 - c. Under the Server Infrastructure heading, select **Java and Process Management**. Select **Process Definition**.
 - d. Under the Additional Properties heading, select **Java virtual machine**.
 - e. Add the following argument to the Generic JVM arguments field, where `<app_server_root>` is the fully qualified path of your application server installation directory. Make sure to use the path separator character appropriate for your operating system.


```
-javaagent:<app_server_root>/optionalLibraries/IBM/wsjava/wsjava.jar
```

Important: The use of generic JVM arguments in the administrative console does not currently support spaces within arguments. If spaces are specified in this field, the server can fail to start. This is more likely to occur in a Windows environment because the default installation path is `C:\Program Files\IBM\WebSphere\AppServer`, which contains a space in the path. To work around this problem in a Windows environment use an abbreviated path name for the `<app_server_root>`. For example, `C:\Progra~1\IBM\WebSphere\AppServer`. On UNIX-type systems a symbolic link can be used to eliminate spaces in the `<app_server_root>`. For example, if the WebSphere Application Server installation path is `/opt/app_server_root/AppServer`, a symbolic link can be created in `/opt` from `<app_server_root>` to `AppServerRoot`, eliminating the space. Then, specify `/opt/AppServerRoot/AppServer` as the `<app_server_root>` in the generic JVM argument.

2. Enable additional trace components and adjust trace file options. You can accomplish this step with `wsadmin` scripting or the administration console. These steps describe how to adjust trace file settings and enable components using the administration console:
 - a. In the navigation pane, select **Troubleshooting**. Click **Logs and Trace**.
 - b. Select the name of the server to trace.
 - c. Under **General Properties**, select **Diagnostic Trace**.
 - d. Make sure **Enable Log** is checked and optionally increase the **Maximum File Size** and **Maximum Number of Historical Files**. Depending on the number of additional trace categories and the trace levels chosen, the trace file can become large.
 - e. Under the Additional Properties heading, select **Change Log Detail Levels**.
 - f. Enable various extended trace categories by specifying one or more trace categories from the following table. An example trace string is: `*=info:JPA=all:openjpa.*=finer:openjpa.kernel=finest`. Extended trace traces at the FINER or FINEST trace levels. The FINEST level includes more detail than FINER. When ALL is specified, extended trace traces at the FINEST level.

Table 5. Trace categories. Back up your server configuration before you enable enhanced tracing.

Category	Relevant trace levels	Description
JPA	OFF, ALL, FINER, FINEST	Adds extended trace to the JPA trace group.
openjpa.*	OFF, ALL, FINER, FINEST	Normal OpenJPA trace in addition to extended trace for all categories in OpenJPA when extended trace is enabled.

Table 5. Trace categories (continued). Back up your server configuration before you enable enhanced tracing.

Category	Relevant trace levels	Description
openjpa.xtrace.*	OFF, ALL, FINER, FINEST	Extended trace for all categories in OpenJPA when extended trace is enabled.
openjpa.xtrace.Jdbc	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA JDBC classes when extended trace is enabled.
openjpa.xtrace.Lib	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA library classes when extended trace is enabled.
openjpa.xtrace.Persist	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA persistence classes when extended trace is enabled.
openjpa.xtrace.Kernel	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA kernel classes when extended trace is enabled.
openjpa.xtrace.General	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA classes not included in the JDBC, Lib, Persist, or Kernel categories when extended trace is enabled.
openjpa.xtrace.ApiSpi	OFF, ALL, FINER, FINEST	Extended trace for public API/SPI interfaces defined for WsJPA, OpenJPA, and JPA when extended trace is enabled.

3. Save the application server configuration and restart the application server.

Results

After you restart the application server, the new trace settings are used.

What to do next

Note: Tracing can degrade performance significantly and should be disabled when not in use. To disable trace, remove the Generic JVM argument and any trace detail levels added for enhanced tracing.

Enabling Enhanced Tracing for JPA in a Java SE environment

In some cases it might be necessary to run a Java Persistence API (JPA) trace from a Java SE environment.

About this task

Enabling enhanced tracing for a Java SE environment

Enhanced trace can also be used gather additional JPA trace information in a Java SE environment. In the WebSphere Application Server environment, the application server takes care of coupling the standard OpenJPA trace with the enhanced trace function to produce a single trace output. This process must be done manually in a Java SE environment. Here are the steps to use enhanced tracing in a Java SE environment:

Procedure

1. Create a logging configuration properties file. The configuration properties file must use the standard java.util.logging configuration file format. The following code is a sample configuration file. The trace

categories defined in the table can also be used in the configuration file. As standard for `java.util.logging` configuration files, trace categories must be suffixed with `.level`.

```
# Sample logger.properties file

# Enable a file handler
handlers = java.util.logging.FileHandler

# Set a trace file pattern - the example writes a file named jpa_jse.log to the current working directory
java.util.logging.FileHandler.pattern = jpa_jse.log

# Configure the basic logging level of the file handler
java.util.logging.FileHandler.level = ALL

# Set the openjp.jdbc.SQL category trace to ALL
openjp.jdbc.SQL.level = ALL

# Set the enhanced General category trace to FINEST
openjp.xtrace.General.level = FINEST

# Set the enhanced Kernel category to trace at FINER
openjp.xtrace.Kernel.level = FINER
```

Table 6. Trace categories. Back up your server configuration before you enable enhanced tracing.

Category	Relevant trace levels	Description
JPA	OFF, ALL, FINER, FINEST	Adds extended trace to the JPA trace group.
openjpa.*	OFF, ALL, FINER, FINEST	Normal OpenJPA trace in addition to extended trace for all categories in OpenJPA when extended trace is enabled.
openjpa.xtrace.*	OFF, ALL, FINER, FINEST	Extended trace for all categories in OpenJPA when extended trace is enabled.
openjpa.xtrace.Jdbc	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA JDBC classes when extended trace is enabled.
openjpa.xtrace.Lib	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA library classes when extended trace is enabled.
openjpa.xtrace.Persist	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA persistence classes when extended trace is enabled.
openjpa.xtrace.Kernel	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA kernel classes when extended trace is enabled.
openjpa.xtrace.General	OFF, ALL, FINER, FINEST	Extended trace for OpenJPA classes not included in the JDBC, Lib, Persist, or Kernel categories when extended trace is enabled.

2. Modify the `persistence.xml` file to use Apache commons logging instead of the default OpenJPA logger. Add this property to your persistence unit:

```
<property name="openjpa.Log" value="commons"/>
```

3. Add the Apache commons logging Java archive (JAR) file to your class path. You can download the JAR file from the Apache website.
4. Add the following argument to the Java virtual machine (JVM), where `<WAS_install_path>` is the fully qualified path of your application server installation directory. Be sure to use the path separator character appropriate for your operating system. This parameter must be specified before the name of the class or JAR file to be used.

```
-javaagent:<WAS_install_path>/optionalLibraries/IBM/wsjava/wsjava.jar
```

5. Add this additional argument to the JVM which specifies the path to your logging configuration file. This option must also be specified before the name of the class or JAR to be used.

```
-Djava.util.Logging.config.file=Logger.properties
```

6. Run your Java SE application. Here is a sample Java SE application invocation with enhanced trace enabled:

```
java
-javaagent:"<WAS_install_path>/optionalLibraries/IBM/wsjava/wsjaxtrace.jar"
-Djava.util.Logging.config.file=Logger.properties
my.JPAAplication
```

Results

Enhanced tracing is now functioning in your Java SE environment.

What to do next

Note: Do not use extended trace agent in combination with the OpenJPA PCEnhancer agent in a Java SE environment. If enhanced tracing is used, the OpenJPA PCEnhancer must be used at compile time. If the enhancer and enhanced trace agents are used together, the results are unpredictable.

Troubleshooting JPA deadlocks and transaction timeouts

Database deadlocks and transaction timeouts are the result of contention between two or more clients attempting to access the same database resource. Deadlock is a special case where a circular blocking condition between two or more clients, each blocked by the others, but no one can proceed. Usually these phenomena are not programming errors. They are caused by business logic accessing database resources in a complex and inter-depending fashions.

About this task

By using the following best practices and strategies, these conditions can be minimized.

Procedure

- Identify the affected database resources that caused the error.
 1. Examine the exceptions (if any) when the error occurs can give you clues on the failing entities that caused in the failing condition.
 2. If the WebSphere Application Server default JPA provider is used, you can enable the product trace group "openjpa.jdbc.SQL=all" to collect the SQL statements and thread information of the database resources that are in question. By collating information, you can find which clients and entities are the artifacts that caused the failing condition.
 3. For more complex usage scenario, you can consult the database documentation for any specialized tool or techniques that can help identify the objects and transaction that are causing the data contention issues.
- When the problems are identified, examine the business logic that uses these resources to determine that their usage does not cause any prolong contention. There is no one prescribed solution to resolve deadlock and timeout problems. It highly depends on the complexity of the business logic and entity relationships. The basic concept to resolve deadlocks and timeouts is to minimize the time of resources being held more than it must and allow other clients to access the same resource. If contention is unavoidable, application must establish means for recovery when these failing conditions occur. The following are strategies that might help you minimize the problem conditions and determine if you can apply one or more to resolve the problem:
 1. Make sure that the business logic does not lock entities more than it must. A JPA application that are mostly read-only should use optimistic lock semantics offered by JPA by default. JPA 2.0 introduces pessimistic locking functions that allow applications to explicitly lock entity pessimistically

on demand. You should optimize the number of pessimistic locks that are applied at any one time. Over locking increases the chance of deadlocks and timeouts, and degrades application concurrency and throughput.

2. Avoid setting data source connection isolation level more restrictive than it requires. Isolation level has the same affect that pessimistic lock semantics do at the connection level. JPA requires the minimum of "read-committed" isolation level to achieve basic data integrity objective. WebSphere Application Server connection manager default isolation level is database-specific. See the Programming interfaces topic for the JDBCConnectionSpec API specification information.
3. Minimize the duration of any active transaction. Extended active transaction has the potential of holding out resources that are required by other clients. Commit any transaction as soon as it has all the resources completed.
4. Optimize business logic to avoid entity inter-dependency. If two sets of business logic uses similar resources, consider updating the resources in the same order to void the classic deadlock circular dependency. This depends on the isolation level being used. It has the same effect as applying implicit lock to the resources as describe in the next strategy.
5. Use pessimistic lock to synchronize concurrent access to common resource. When the previous strategies do not apply to your situation, pessimistic locking is an alternative to synchronize access to common resources used by different concurrent clients. The application can use JPA 2.0 pessimistic locking functions to block other clients from accessing the common resource until the transaction is committed or rolled back. Increase of pessimistic lock usage can affect concurrency and throughput. WebSphere Application Server JPA Access Intent also provides another alternative to lock data resources based on EJB invocation or user-defined task name.
6. Handle and recover from deadlock and timeout exception. Most database servers have a built-in deadlock prevention mechanism. When it detects a deadlock condition, the typical strategy used by the database is to cancel one of the requests and let the other one succeed. However the request that is terminated is database-specific and non-deterministic. If an application allows, it is a good practice to recover the deadlock condition and attempt to retry the operation.

Results

For more information, review the Apache OpenJPA User Guide.

Chapter 9. Troubleshooting Messaging resources

This page provides a starting point for finding information about the use of asynchronous messaging resources for enterprise applications with WebSphere Application Server.

WebSphere Application Server supports asynchronous messaging based on the Java Message Service (JMS) and the Java EE Connector Architecture (JCA) specifications, which provide a common way for Java programs (clients and Java EE applications) to create, send, receive, and read asynchronous requests, as messages.

JMS support enables applications to exchange messages asynchronously with other JMS clients by using JMS destinations (queues or topics). Some messaging providers also allow WebSphere Application Server applications to use JMS support to exchange messages asynchronously with non-JMS applications; for example, WebSphere Application Server applications often need to exchange messages with WebSphere MQ applications. Applications can explicitly poll for messages from JMS destinations, or they can use message-driven beans to automatically retrieve messages from JMS destinations without explicitly polling for messages.

WebSphere Application Server supports the following messaging providers:

- The WebSphere Application Server default messaging provider (which uses service integration as the provider).
- The WebSphere MQ messaging provider (which uses your WebSphere MQ system as the provider).
- Third-party messaging providers that implement either a JCA Version 1.5 resource adapter or the ASF component of the JMS Version 1.0.2 specification.

Troubleshooting messaging

Use this overview task to help resolve a problem that you think is related to JMS messaging in WebSphere Application Server.

About this task

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

To identify and resolve problems that you think are related to messaging, you can use the standard WebSphere Application Server troubleshooting facilities. Some problems are specific to a particular messaging provider; that is the default messaging provider (service integration), the WebSphere MQ messaging provider, the V5 default messaging provider or a third-party messaging provider.

Procedure

1. Check for error messages about messaging:
 - a. Check for error messages that indicate a problem with JMS resources.
Check in the SYSPRINT or SYSOUT log.
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 78.

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 technotes web page at <http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP+SSCMGN&dc=DB520+DB560&dtm> for specific problems and solutions.

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 information about configuring JMS resources for each of the available messaging providers, see the following topics:

- Configuring resources for the default messaging provider
- Configuring JMS resources for the WebSphere MQ messaging provider
- Managing messaging with a third-party JCA 1.5-compliant messaging provider
- Configuring JMS resources for a third-party non-JCA messaging provider
- Configuring Version 5 default messaging resources

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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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.

To help you identify and resolve problems with messaging, you can use the WebSphere Application Server trace and logging facilities as described in Setting up component trace (CTRACE).

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

- “Troubleshooting messaging” on page 71
- “Troubleshooting message-driven beans” on page 78

- “Troubleshooting service integration technologies” on page 153
- Chapter 19, “Default messaging provider: Troubleshooting tips,” on page 205

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, by 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.

- “WebSphere MQ resource adapter configuration is not automatically updated and requires manual maintenance”
- “java.lang.ClassNotFoundException exceptions occur when you install a fix pack”
- “Messages from WebSphere MQ for z/OS are not being consumed by JMS applications that are deployed into WebSphere Application Server and that use connection factories or activation specifications” on page 74
- “A JMS application can no longer send or receive messages, or a destination becomes full and can no longer receive messages” on page 74
- “javax.jms.JMSEException: MQJMS2008: failed to open MQ queue in JVM log ” on page 74
- “An MDB listener fails to start” on page 74
- “Problems running JMS applications with security enabled” on page 75
- “Server memory consumption and java.lang.OutOfMemoryError exception when processing JMS messages” on page 75
- “TopicConnectionFactory attributes clash error when you use “Basic” WebSphere MQ broker (MAOC SupportPac broker)” on page 76
- “WSEC5061E: The SOAP Body is not signed exception is issued when running a secured web services application that uses JMS transport and WebSphere MQ” on page 76
- “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 an application server on a later version of the product” on page 76
- “When you use WebSphere MQ as an external JMS provider, messages sent within a user-managed transaction arrive before the transaction commits” on page 77
- “For the WebSphere MQ messaging provider, channel framework messages appear during server startup” on page 77

WebSphere MQ resource adapter configuration is not automatically updated and requires manual maintenance

Normally the WebSphere MQ resource adapter is automatically updated when you apply a WebSphere Application Server fix pack. However, if you have manually updated the WebSphere MQ resource adapter on some nodes in your environment, applying a fix pack does not automatically update the resource adapter that is used by servers on those nodes.

To resolve this issue, see Maintaining the WebSphere MQ resource adapter.

java.lang.ClassNotFoundException exceptions occur when you install a fix pack

If, when installing a fix pack you see the following message, follow the instructions in Maintaining the WebSphere MQ resource adapter to try and resolve the problem:

```
J2CA0043E: An exception occurred while trying to instantiate a ResourceAdapter
JavaBean instance for the installed ResourceAdapter defined by key #removed#
```

Messages from WebSphere MQ for z/OS are not being consumed by JMS applications that are deployed into WebSphere Application Server and that use connection factories or activation specifications

For more information about how to configure the **Provider version** property, see any of the following topics:

- Configuring an activation specification for the WebSphere MQ messaging provider.
- Configuring a unified connection factory for the WebSphere MQ messaging provider.
- Configuring a queue connection factory for the WebSphere MQ messaging provider.
- Configuring a topic connection factory for the WebSphere MQ messaging provider.

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: `../ae/AppToSIBRefs_DetailForm.dita`.
- For a view of the applications and JMS resources for a given default messaging provider destination, see the following panel: `../ae/AppsFromSIBRefs_DetailForm.dita`.

javax.jms.JMSEException: 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 > Server Types > Version 5 JMS servers**
3. Add the queue name to the list.
4. Save the changes and restart the server.

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 the user ID used to start the MDB listener is appropriately authorized. For more information, see “Problems running JMS applications with security enabled” on page 75.

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. Alternatively, when running in *bindings transport mode* you can exploit the connector thread identity support.

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 *Securing messaging* 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 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 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 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 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 > Server Types > WebSphere application servers > server_name > Java and Process Management > Process Definition > Java Virtual Machine**, then configure the amount of memory available to the application server 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 you use "Basic" WebSphere MQ broker (MAOC 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 (MAOC 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 (MAOC SupportPac broker).

“WSEC5061E: The SOAP Body is not signed” exception is issued when running a secured web services application that uses 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.S 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 complete 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 an application server on a later version of the product

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 an application server on a later version of the product, 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.JMSEException: MQJMS1006: invalid value for tempQPrefix:
```

This problem occurs when the application client is using WebSphere MQ JMS client CSD 04 JAR files. The later version of the product 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.

For the WebSphere MQ messaging provider, channel framework messages appear during server startup

The following message might be displayed several times in the control region adjunct process during server startup, even though the connection succeeds on subsequent retries. This message is issued because of the asynchronous way in which the z/OS TCP proxy channel starts and does not indicate that any error has occurred.

```
Trace: 2009/06/17 08:24:41.434 01 t=9C6B58 c=UNK key=P8 (00000011)
Description: Log Java Message
Message: CHF0030E: Error starting chain_InboundTCPProxyBridgeService because
of exception com.ibm.wsspi.channel.framework.exception.RetryableChannelException:
An exception was thrown when attempting to start the TCPProxyChannel
com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl
```

These messages might be accompanied by First Failure Data Capture (FFDC) output similar to the following example:

```
Exception = com.ibm.wsspi.channel.framework.exception.RetryableChannelException
Source = com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.startChainInternal
probeid = 2577
Stack Dump = com.ibm.wsspi.channel.framework.exception.RetryableChannelException: An exception was thrown when at
  at com.ibm.ws.tcpchannelproxy.jfap.impl.TCPProxyInboundChannel.
start(TCPProxyInboundChannel.java:153)
  at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.
startChainInternal(ChannelFrameworkImpl.java:1410)
  at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.
startChainInternal(ChannelFrameworkImpl.java:2863)
  at com.ibm.ws.channel.framework.impl.WSChannelFrameworkImpl.
startChainInternal(WSChannelFrameworkImpl.java:960)
  at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.
startChainInternal(ChannelFrameworkImpl.java:2794)
  at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.
startChain(ChannelFrameworkImpl.java:2779)
  at com.ibm.ws.runtime.component.ChannelFrameworkServiceImpl.
```

```

startChain(ChannelFrameworkServiceImpl.java:666)
    at com.ibm.ws.sib.jfapchannel.framework.impl.ChannelFrameworkReference$TCPProxy
BridgeServiceInboundChainStartupRunnable.run(ChannelFrameworkReference.java:1641)
    at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1550)
Caused by: com.ibm.ws.tcpchannelproxy.jfap.NotYetInitializedException:
Server is not yet initialized
    at com.ibm.ws.tcpchannelproxy.jfap.TCPProxyBridgeServicesImpl.
startListening(TCPProxyBridgeServicesImpl.java:558)
    at com.ibm.ws.tcpchannelproxy.jfap.impl.TCPProxyInboundChannel.
start(TCPProxyInboundChannel.java:131)
    ... 8 more

```

Eventually the following message should be displayed indicating that the z/OS TCP proxy channel has started up correctly:

```

Trace: 2009/06/17 08:24:51.449 01 t=9C6B58 c=UNK key=P8 (13007002)
  ThreadId: 00000003
  FunctionName: com.ibm.ws.channel.framework.impl.WSChannelFrameworkImpl
  SourceId: com.ibm.ws.channel.framework.impl.WSChannelFrameworkImpl
  Category: AUDIT
  ExtendedMessage: BB000222I: CHFW0019I: The Transport Channel Service has started
chain_InboundTCPProxyBridgeService.

```

WebSphere MQ connection and queue connection factory creation errors

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

The following exception might 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 WebSphere 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

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance

Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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.

Procedure

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 SYSPRINT or SYSOUT log.

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: `../ae/AppToSIBRefs_DetailForm.dita`.
 - For a view of the message-driven beans and JMS resources for a given default messaging provider destination, see the following panel: `../ae/AppsFromSIBRefs_DetailForm.dita`.
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 website, 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 website.
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 that 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.
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 might 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.

Procedure

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

Chapter 10. Troubleshooting Naming and directory

This page provides a starting point for finding information about naming support. Naming includes both server-side and client-side components. The server-side component is a Common Object Request Broker Architecture (CORBA) naming service (CosNaming). The client-side component is a Java™ Naming and Directory Interface (JNDI) service provider. JNDI is a core component in the Java Platform, Enterprise Edition (Java EE) programming model.

The WebSphere® JNDI service provider can be used to interoperate with any CosNaming name server implementation. Yet WebSphere name servers implement an extension to CosNaming, and the JNDI service provider uses those WebSphere extensions to provide greater capability than CosNaming alone. Some added capabilities are binding and looking up of non-CORBA objects.

Java EE applications use the JNDI service provider supported by WebSphere Application Server to obtain references to objects related to server applications, such as enterprise bean (EJB) homes, which have been bound into a CosNaming name space.

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.

Procedure

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 82 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:

- 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.

Remember: 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.

- 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" in the topic on application access problems.
- 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, stand-alone 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 " on page 83
- "CannotInstantiateObjectException from JNDI lookup operation " on page 84
- "Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred " on page 84
- "OperationNotSupportedException from JNDI Context operation" on page 84
- "WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound " on page 85
- "ConfigurationException from "new InitialContext" operation or from a JNDI Context operation with a URL name" on page 85
- "ServiceUnavailableException from "new InitialContext" operation" on page 85
- "CommunicationException thrown from a "new InitialContext" operation" on page 86
- 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 stand-alone 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
```

Restriction: 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.

View the logs of the server hosting the target resource.

- 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 89 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.

Procedure

- Run the tool from a command line.

Enter the dumpNameSpace command from the *app_server_root/bin* directory.

```
dumpNameSpace.sh [-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.

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/ur1  javax.naming.Context
31 (top)/nodes/outpost/servers/server1/ur1/CatalogDAOURL
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.

gotcha: If your system has more than one IP address, you need to modify the order in which the system configured name lookup mechanism processes during the running of the dumpNameSpace tool. You need to use the `-Dsun.net.spi.nameservice.provider.1=dns,sun` setting, a Java option, to ensure that the dumpNameSpace tool works successfully on systems which have more than one IP address.

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.sh [[-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 servers:

Table 7. -root option descriptions for product servers. The root context provides the initial context for the dump.

-root option	Description
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.
applications	Dumps the tree starting at the applications root context.

For all WebSphere Application Server and other name servers:

Table 8. -root option description for product and non-product servers. The root context provides the initial context for the dump.

-root option	Description
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* }

Table 9. -format option descriptions. Options include jndi and ins.

-format option	Description
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 }

Table 10. *-report option descriptions. Options include short and long.*

-report option	Description
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:

Table 11. *dumpNameSpace tool return codes. A return code of 0 indicates no error. Return codes of 1 through 3 indicate an error condition.*

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.
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.

Procedure

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.sh
```

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 93. 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 11. Troubleshooting Object Request Broker (ORB)

This page provides a starting point for finding information about the Object Request Broker (ORB). The product uses an ORB to manage communication between client applications and server applications as well as among product components. These Java Platform, Enterprise Edition (Java EE) standard services are relevant to the ORB: Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP) and Java Interface Definition Language (Java IDL).

The ORB provides a framework for clients to locate objects in the network and call operations on those objects as though the remote objects were located in the same running process as the client, providing location transparency.

Troubleshooting Object Request Brokers

Object request broker component troubleshooting tips

The following topics might help you diagnose problems with the Object request broker (ORB) component.

Enabling tracing for the Object Request Broker component

The object request broker (ORB) service is one of the product run time services. Tracing messages that are sent and received by the ORB is a useful starting point for troubleshooting the ORB service. You can selectively enable or disable tracing of ORB messages for each server in a product installation, and for each application client.

For instructions on how to set trace controls so that tracing occurs for the ORB subcomponent, read the topic on setting trace controls for IBM support.

Log files and messages associated with Object Request Broker

For a summary of how messages get routed in the product, read the topic on managing operator message routing.

Java packages containing the Object Request Broker service

The ORB service resides in the following Java packages:

com.ibm.CORBA.*

This package provides the mapping of the IBM CORBA APIs to the Java programming language, including the class ORB.

com.ibm.rmi.*

This package provides the IBM Remote Method Invocation (RMI) APIs that are used to establish remote communication between programs written in the Java programming language.

com.ibm.ws.orb.*

This package provides the APIs that are used to specify configuration settings for the ORB.

com.ibm.ws.orbimpl.*

This package provides the IBM implementation classes for the ORB.

com.ibm.ws390.orb.*

This package provides z/OS-only classes.

com.ibm.ws390.channel.ziop.*

This package contains portability APIs for z/OS IIOP transport channels.

com.ibm.ws390.ziop.*

This package contains portability APIs for the z/OS IIOp runtime.

org.omg.CORBA.*

This package provides the mapping of the OMG CORBA APIs to the Java programming language, including the class ORB.

javax.rmi.CORBA.*

This package contains portability APIs for the RMI-IIOP runtime.

JAR files that contain the previously mentioned packages include:

- *app_server_root/java/lib/ibmorb.jar*, which contains the base ORB classes `com.ibm.CORBA.*`, `com.ibm.rmi.*`, `javax.rmi.CORBA.*`, and `org.omg.CORBA.*`
- *app_server_root/plugins/com.ibm.ws.runtime.jar*, which contains the extension classes `com.ibm.ws.orb.*`, `com.ibm.ws.orbimpl.*`, and `com.ibm.CORBA.services.*`.
- *app_server_root/plugins/com.ibm.ws.runtime.ws390.jar*, which contains the `com.ibm.ws390.*` classes.
- *app_server_root/lib/bootstrap.jar* and *app_server_root/lib/bootstrapws390.jar*, which include some ORB related classes.

Tools used with Object Request Broker

The tools used to compile Java remote interfaces to generate language bindings used by the ORB at runtime reside in the following APIs:

- `com.ibm.tools.rmic.*`
- `com.ibm.idl.*`

The JAR file that contains these APIs is *app_server_root/java/lib/tools.jar*.

Object Request Broker properties

The ORB service requires a number of ORB properties for correct operation. It is not necessary for most users to modify these properties, and only the system administrator should modify them when required. Consult IBM Support personnel for assistance. The properties reside in the properties file, located at *app_server_root/properties/orb.properties*.

If none of these steps fixes your problem, check the Support page to see if the problem has been identified and documented. The Support page contains hints and tips, technotes, and descriptions of available fixes.

Before opening a problem report, collect the information that Support needs to resolve problems. The topic on help with troubleshooting includes documents and tools that can help you gather this information.

Chapter 12. Troubleshooting OSGi applications

This page provides a starting point for finding out how to troubleshoot OSGi applications.

Chapter 13. Troubleshooting 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. This function was deprecated in WebSphere Application Server 7.0. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

Procedure

- 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
- Authorization provider troubleshooting tips

Security components troubleshooting tips

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

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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

- What to look at and what to look for “Using SDSF” on page 100.
- “General approach for troubleshooting security-related issues” on page 100 to isolating and resolving security problems.
- When and how to “Trace security” on page 107. Setting up component trace (CTRACE) is a topic that will aid your tracing activity.

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

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

For an overview of WebSphere Application Server security components such as Secure Authentication Services for z/OS (z/SAS) and how they work, refer to the *Securing applications and their environment* PDF book..

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

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:

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.

```
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: BB0J0077I 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.ws.390.orb.CommonBridge
Category: AUDIT
ExtendedMessage: BB0J0077I 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: SECJ0136I: Custom
Registry:com.ibm.ws.security.registry.z0S.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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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: BB000222I: 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 the Tracing and logging configuration article.

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 are passed from the controller to the servant. Web requests are mostly ignored by the controller. As a result, EJB requests are first processed and authenticated by the zSAS or Common Security Interoperability Version 2 (CSIv2) layers of security. Authorization is done by the servant. If an authentication failure occurs, the zSAS type level of tracing must be turned on to diagnose the problem. Other problems can be diagnosed using the WebSphere Application Server component tracing (CTRACE) facility.

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.

In z/OS, two variations of SSL are used. To determine the cause of an SSL problem on z/OS, you have to be aware of what protocol is being used.

- System SSL is used by the Internet Inter-ORB Protocol (IIOP) and HTTPS protocols
- Java Secure Socket Extension (JSSE) is used by all other protocols, for example, SOAP.
- System SSL requests are handled in the controller and are used by z/SAS and CSIv2 security.
- SJSSE is predominately used by the servant, but cases exist where JSSE is used in the controller as well.

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.
- Recreate the problem.

The SYSOUT data set for the region's started task contains the JSSE trace. Using SDSF, this trace is similar to the following:

```
JSSSEContext: handleConnection[Socket
[addr=boss0106.plex1.12.ibm.com/9.38.48.108,port=2139,localport=8878]
JSSSEContext: handleConnection[Socket
[addr=boss0106.plex1.12.ibm.com/9.38.48.108,port=2140,localport=8878]
TrustManagerFactoryImpl: trustStore is :
/WebSphere/V6R10M0/AppServer/etc/DummyServerTrustFile.jks
TrustManagerFactoryImpl: trustStore type is : JKS
TrustManagerFactoryImpl: init truststore
JSSSEContext: handleConnection[Socket
[addr=boss0106.plex1.12.ibm.com/9.38.48.108,port=2142,localport=8878]
KeyManagerFactoryImpl: keyStore is :
/WebSphere/V6R10M0/AppServer/etc/DummyServerKeyFile.jks
KeyManagerFactoryImpl: keyStore type is : JKS
KeyManagerFactoryImpl: init keystore
KeyManagerFactoryImpl: init keystore
JSSSEContext: handleConnection[Socket
[addr=boss0106.plex1.12.ibm.com/9.38.48.108,port=2143,localport=8878]
JSSSEContext: handleSession[Socket
[addr=BOSSXXXX.PLEX1.L2.IBM.COM/9.38.48.108,port=8879,localport=2145]
JSSSEContext: confirmPeerCertificate
[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/9.38.48.108,port=8879,
localport=2145]]
X509TrustManagerImpl: checkServerTrusted
X509TrustManagerImpl: Certificate [
[
```

```

Version: V3
Subject: CN=jserver, OU=SWG, O=IBM, C=US
Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0 Key: IBMJCE RSA Public Key:
modulus:
1009499669223950907479682875611853910756836956631388995538950668
6622953008589748001058216362638201577071902071311277365773252660799
128781182947273802312699983556527878615792292244995317112436562491
489904381884265119355037731265408654007388863303101746314438337601
264540735679944205391693242921331551342247891
public exponent:
65537
0 Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
To: Thu Mar 17 20:08:18 GMT 2005]
Issuer: CN=jserver, OU=SWG, O=IBM, C=US
SerialNumber: [ 3d1387b2 ]
0]
Algorithm: [MD5withRSA]
Signature:
0000: 54 DC B5 FA 64 C9 CD FE B3 EF 15 22 3D D0 20 31 T...d....."=. 1
0010: 99 F7 A7 86 F9 4C 82 9F 6E 4B 7B 47 18 2E C6 25 .....L..nK.G...%
0020: 5B B2 98 78 D8 76 5C 82 07 95 DD B8 44 62 02 62 [...x.v\.....Db.b
0030: 60 2A 0A 6D 4F B9 0A 98 14 27 E9 BB 1A 84 8A D1 ~*.mO.....'.....
0040: C2 22 AF 70 9E A5 DF A2 FD 57 37 CE 3A 63 1B EB ."p.....W7.:c..
0050: E8 91 98 9D 7B 21 4A B5 2C 94 FC A9 30 C2 74 72 .....!J,....0.tr
0060: 95 01 54 B1 29 E7 F8 9E 6D F3 B5 D7 B7 D2 9E 9B ..T.)...m.....
0070: 85 D8 E4 CF C2 D5 3B 64 F0 07 17 9E 1E B9 2F 79 .....;d...../y
0]
X509TrustManagerImpl: Certificate [
[
Version: V3
Subject: CN=jserver, OU=SWG, O=IBM, C=US
Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0 Key: IBMJCE RSA Public Key:
modulus:
100949966922395090747968287561185391075683695663138899553895066866
22953008589748001058216362638201577071902071311277365773252660799
1287811829472738023126999835565278786157922922449953171124365624914
89904381884265119355037731265408654007388863303101746314438337601
264540735679944205391693242921331551342247891
public exponent:
65537
0 Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
To: Thu Mar 17 20:08:18 GMT 2005]
Issuer: CN=jserver, OU=SWG, O=IBM, C=US
SerialNumber: [ 3d1387b2 ]
0]
Algorithm: [MD5withRSA]
Signature:
0000: 54 DC B5 FA 64 C9 CD FE B3 EF 15 22 3D D0 20 31 T...d....."=. 1
0010: 99 F7 A7 86 F9 4C 82 9F 6E 4B 7B 47 18 2E C6 25 .....L..nK.G...%
0020: 5B B2 98 78 D8 76 5C 82 07 95 DD B8 44 62 02 62 [...x.v\.....Db.b
0030: 60 2A 0A 6D 4F B9 0A 98 14 27 E9 BB 1A 84 8A D1 ~*.mO.....'.....
0040: C2 22 AF 70 9E A5 DF A2 FD 57 37 CE 3A 63 1B EB ."p.....W7.:c..
0050: E8 91 98 9D 7B 21 4A B5 2C 94 FC A9 30 C2 74 72 .....!J,....0.tr
0060: 95 01 54 B1 29 E7 F8 9E 6D F3 B5 D7 B7 D2 9E 9B ..T.)...m.....
0070: 85 D8 E4 CF C2 D5 3B 64 F0 07 17 9E 1E B9 2F 79 .....;d...../y
0]
JSSSEContext: handleConnection[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=2144,localport=8878]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2145]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2146]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2147]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2148]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2149]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2150]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2151]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2152]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2153]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2154]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2155]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2156]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2157]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2158]]
JSSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2159]]

```

```

JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2160]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2161]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2162]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2163]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2164]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2165]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2166]]

JSSEContext: handleSession[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
JSSEContext: confirmPeerCertificate[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
X509TrustManagerImpl: checkServerTrusted
X509TrustManagerImpl: Certificate [
[
  Version: V3
  Subject: CN=WAS z/OS Deployment Manager, O=IBM
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
0 Key: IBMJCE RSA Public Key:
modulus:
12840948267119651469312486548020957441946413494498370439558603901582589
8755033448419534105183133064366466828741516428176579440511007
6258795528749232737808897160958348495006972731464152299032614592135114
19361539962555997136085140591098259345625853617389396340664766
649957749527841107121590352429348634287031501
public exponent:
65537
0 Validity: [From: Fri Jul 25 05:00:00 GMT 2003,
To: Mon Jul 26 04:59:59 GMT 2004]
  Issuer: CN=WAS CertAuth, C=US
  SerialNumber: [ 02 ]
0Certificate Extensions: 3
[1]: ObjectID: 2.16.840.1.113730.1.13 Criticality=false
Extension unknown: DER encoded OCTET string =
0000: 04 3C 13 3A 47 65 6E 65 72 61 74 65 64 20 62 79 .<.:Generated by
0010: 20 74 68 65 20 53 65 63 75 72 65 57 61 79 20 53 the SecureWay S
0020: 65 63 75 72 69 74 79 20 53 65 72 76 65 72 20 66 ecurity Server f
0030: 6F 72 20 7A 2F 4F 53 20 28 52 41 43 46 29 or z/OS (RACF)
-[2]: ObjectID: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 05 6A CD 7F AE AF 89 78 99 A8 F1 5B 64 8B 9F AF .j.....x...[d...
0010: 73 1B 58 65 s.Xe
]
]
0[3]: ObjectID: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 7E D1 7B 17 74 D3 AD D1 7D D8 F8 33 85 19 04 F8 ....t.....3....
0010: 36 51 57 16 6QW.
]
]
0]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 73 0D FC E1 8A B3 42 E1 04 73 72 B1 C6 C9 87 54 s.....B..sr....T
0010: 87 57 02 FA 41 32 D8 B0 39 09 86 CB 6B 03 B6 F9 .W..A2..9...k...
0020: 62 8D 95 36 56 0E D4 D2 F7 7A 8D 4B FB 0B FD 91 b..6V....z.K....
0030: 89 A8 08 41 30 E2 27 DC 15 5F 2C F4 CD 2F 6B 8E ..A0.'..._./k.
0040: 21 2A 88 53 46 27 68 9B 55 14 38 8E 1F 50 95 BC !*.SF'h.U.8..P..
0050: A8 46 F6 68 97 9E 7B 65 9E E8 A7 34 B2 C8 63 CF .F.h...e...4..c.
0060: 73 C8 4E 25 0A EF C5 8F 04 A4 EB 8C CC 33 84 26 s.N%.....3.&
0070: 5D FD 7C AD 7B 02 13 5A 86 A1 89 93 1E A4 93 63 ].....Z.....c
0]
X509TrustManagerImpl: Certificate [
[
  Version: V3
  Subject: CN=WAS CertAuth, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
0 Key: IBMJCE RSA Public Key:
modulus:
1167408593733331602218385578183389496484587418638676352829560040529918
40558681208199977833401609895748222369066230329785148883251144
2382911186804921983976695395381692334250582278359056431484427844566504
41491799952592864895242987037929408453455627552772317382077015
828713585220212502839546496071839496308430393
public exponent:
65537
0 Validity: [From: Fri Jul 25 05:00:00 GMT 2003,
To: Sat Jul 24 04:59:59 GMT 2010]
  Issuer: CN=WAS CertAuth, C=US
  SerialNumber: [ 0 ]
0Certificate Extensions: 4

```

```

[1]: ObjectId: 2.16.840.1.113730.1.13 Criticality=false
Extension unknown: DER encoded OCTET string =
0000: 04 3C 13 3A 47 65 6E 65 72 61 74 65 64 20 62 79  .<.:Generated by
0010: 20 74 68 65 20 53 65 63 75 72 65 57 61 79 20 53  the SecureWay S
0020: 65 63 75 72 69 74 79 20 53 65 72 76 65 72 20 66  ecurity Server f
0030: 6F 72 20 7A 2F 4F 53 20 28 52 41 43 46 29  or z/OS (RACF)
-[2]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 7E D1 7B 17 74 D3 AD D1 7D D8 F8 33 85 19 04 F8  ....t.....3....
0010: 36 51 57 16 6QW.
]
]
0[3]: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
Key_CertSign
Crl_Sign
]
0[4]: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
CA:true
PathLen:2147483647
]
0]
Algorithm: [SHA1withRSA]
Signature:
0000: 43 88 AB 19 5D 00 54 57 5E 96 FA 85 CE 88 4A BF C...].TW^.....J.
0010: 6E CB 89 4C 56 BE EF E6 8D 2D 74 B5 83 1A EF 9C n...LV....-t.....
0020: B3 82 F2 16 84 FA 5C 50 53 2A B4 FD EB 27 98 5D ..... \PS*...'.]
0030: 43 48 D3 74 85 21 D1 E1 F2 63 9E FB 58 2A F3 6A CH.t.!...c.X*.j
0040: 44 D2 F5 7D B2 55 B9 5E 32 11 78 B6 34 8E 4B 1D D...U.^2.x.4.K.
0050: F3 82 1D C1 5F 7B 3F AD C9 29 FA FF D1 D1 13 2C ..._.?..).....
0060: 57 F7 7B 51 02 99 6F ED 54 E1 51 34 B8 51 BE 97 W..Q..o.T.Q4.Q..
0070: 30 AC 4F 89 AB AA 8A B2 E1 40 89 2E 18 C7 0E 15 0.0.....@.....
0]
JSSContext: handleConnection[Socket[addr=boss0106.plex1.12.ibm.com
/9.38.48.108,port=9443,localport=2167]]
JSSContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2168]]

JSSContext: handleConnection[Socket[addr=boss0106.plex1.12.ibm.com
/9.38.48.108,port=2235,localport=8878]]
JSSContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2236]]
JSSContext: handleSession[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8880,localport=2238]]
JSSContext: confirmPeerCertificate[Socket
[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8880,localport=2238]]
X509TrustManagerImpl: checkServerTrusted
X509TrustManagerImpl: Certificate [

[
Version: V3
Subject: CN=jsrserver, OU=SWG, O=IBM, C=US
Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0 Key: IBMJCE RSA Public Key:
modulus:
10094996692239509074796828756118539107568369566313889955538950668622953
008589748001058216362638201577071902071311277365773252660799
1287811829472738023126999835565278786157922922449953171124365624914
89904381884265119355037731265408654007388863303101746314438337601
264540735679944205391693242921331551342247891
public exponent:
65537
0 Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
To: Thu Mar 17 20:08:18 GMT 2005]
Issuer: CN=jsrserver, OU=SWG, O=IBM, C=US
SerialNumber: [ 3d1387b2 ]
0]
Algorithm: [MD5withRSA]
Signature:
0000: 54 DC B5 FA 64 C9 CD FE B3 EF 15 22 3D D0 20 31 T...d....."=. 1
0010: 99 F7 A7 86 F9 4C 82 9F 6E 4B 78 47 18 2E C6 25 .....L.nk.G...%
0020: 5B B2 98 78 D8 76 5C 82 07 95 DD B8 44 62 02 62 [...x.v\.....Db.b
0030: 60 2A 0A 6D 4F B9 0A 98 14 27 E9 BB 1A 84 8A D1 ~*.mO....'.....
0040: C2 22 AF 70 9E A5 DF A2 FD 57 37 CE 3A 63 1B EB ".p.....W7.:c.
0050: E8 91 98 9D 7B 21 4A B5 2C 94 FC A9 30 C2 74 72 .....!J...0.tr
0060: 95 01 54 B1 29 E7 F8 9E 6D F3 B5 D7 B7 D2 9E 9B ..T.)...m.....
0070: 85 D8 E4 CF C2 D5 3B 64 F0 07 17 9E 1E B9 2F 79 .....;d...../y
0]
X509TrustManagerImpl: Certificate [
[
Version: V3
Subject: CN=jsrserver, OU=SWG, O=IBM, C=US
Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0 Key: IBMJCE RSA Public Key:
modulus:
100949966922395090747968287561185391075683695663138899555389506
686622953008589748001058216362638201577071902071311277365773252660799

```

```

12878118294727380231269998355652787861579229224499531711243656249
1489904381884265119355037731265408654007388863303101746314438337601
264540735679944205391693242921331551342247891
public exponent:
65537
0 Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
             To: Thu Mar 17 20:08:18 GMT 2005]
  Issuer: CN=jserver, OU=SWG, O=IBM, C=US
  SerialNumber: [ 3d1387b2 ]
0]
  Algorithm: [MD5withRSA]
  Signature:
0000: 54 DC B5 FA 64 C9 CD FE B3 EF 15 22 3D D0 20 31 T...d....."=. 1
0010: 99 F7 A7 86 F9 4C 82 9F 6E 4B 7B 47 18 2E C6 25 .....L..nK.G...%
0020: 5B B2 9B 78 D8 76 5C 82 07 95 DD B8 44 62 02 62 [..x.v\....Db.b
0030: 60 2A 0A 6D 4F B9 0A 98 14 27 E9 BB 1A 84 8A D1 ~*.mO....'.....
0040: C2 22 AF 70 9E A5 DF A2 FD 57 37 CE 3A 63 1B EB .."p.....W7.:c..
0050: E8 91 98 9D 7B 21 4A B5 2C 94 FC A9 30 C2 74 72 .....!J,...0.tr
0060: 95 01 54 B1 29 E7 F8 9E 6D F3 B5 D7 B7 D2 9E 9B ..T)...m...../
0070: 85 D8 E4 CF C2 D5 3B 64 F0 07 17 9E 1E B9 2F 79 .....;d...../y
0]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2238]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2239]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2240]]
JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2241]]

```

Trace security

The classes that implement WebSphere Application Server security are:

- com.ibm.ws.security.*
- com.ibm.websphere.security.*
- com.ibm.WebSphereSecurityImpl.*
- com.ibm.ws.wim.* for tracing with a Virtual Member Manager (VMM) repository

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 z/SAS Trace Settings

Situations arise where reviewing trace for the CSiv2 and z/SAS authentication protocols can assist in troubleshooting difficult problems. This section describes how to enable to CSiv2 and z/SAS trace.

Enabling Client-Side CSiv2 and z/SAS Trace

To enable CSiv2 and z/SAS trace on a pure client, the following steps need to be taken:

- Edit the file TraceSettings.properties in the /WebSphere/AppServer/properties directory.
- In this file, change traceFileName= to point to the path in which you want the output file created. Make sure you put a double backslash (\\) between each subdirectory. For example, traceFileName=c:\\WebSphere\\AppServer\\logs\\sas_client.log
- In 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 z/SAS Trace

To enable z/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 administrative console.

- Typically it is best to also trace the authorization security runtime in addition to the authentication protocol run time. 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. To set these properties, click **Servers > Server Types > WebSphere application servers > server_name > Container services > ORB service**. Select the **ORB tracing** option. This option sets two properties that you had to manually set in a previous release. For more information, see the Object Request Broker service settings topic.

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)
```

Table 12. CORBA minor codes after running a security-related request such as authentication. The following table shows the CORBA minor codes which a client can expect to receive after running a security-related request such as authentication. The client can be either a stand-alone client or a server acting as a client. It also includes the CORBA exception type that the minor code would appear in.

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA .*)	Minor code description	Retry performed by stand-alone client (when authenticationRetryEnabled = true)	Retry performed
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	Yes
InterceptLocateException	494210B8	INTERNAL	This indicates a problem when processing an incoming locate request.	No	No
InvalidUserId	49424301	NO_PERMISSION	This code occurs when the registry returns bad user ID.	Yes	No
InvalidPassword	49424302	NO_PERMISSION	This code occurs when the registry returns a bad password.	Yes	No
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).	Yes
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	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).	Yes
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).	Yes
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).	No
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	Yes
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	Yes

Table 12. CORBA minor codes after running a security-related request such as authentication (continued). The following table shows the CORBA minor codes which a client can expect to receive after running a security-related request such as authentication. The client can be either a stand-alone client or a server acting as a client. It also includes the CORBA exception type that the minor code would appear in.

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA.*)	Minor code description	Retry performed by stand-alone client (when authenticationRetryEnabled = true)	Retry performed by s
SessionRejected	4942430A	NO_PERMISSION	This indicates that the session referenced by the client has been previously rejected by the server.	Yes	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	No
InvalidIdentityToken	4942430C	NO_PERMISSION	This error indicates that identity cannot be obtained from the identity token when Identity Assertion is enabled.	No	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	No
InvalidMessage	4942430E	NO_PERMISSION	This indicates that the CSv2 message format is not valid for the receiving server.	No	No
MappingFailed	4942430F	NO_PERMISSION	This indicates an error occurred mapping an inbound subject using the RMI Inbound system login configuration.	No	No
RevokedSecurityName	49424310	NO_PERMISSION	This indicates that the user id is revoked.	Yes	No
ExpiredPassword	49424311	NO_PERMISSION	This indicates that the password is expired.	Yes	No
AuthenticationNotSupported	49421090	NO_PERMISSION	This error occurs when a mechanism does not support authentication (very rare).	No	No
InvalidSecurityMechanism	49421091	NO_PERMISSION	This is used to indicate that the specified security mechanism is not known.	No	No
CredentialNotAvailable	49421092	NO_PERMISSION	This indicates a credential is not available when it is required.	No	No
SecurityMechanismNotSupported	49421093	NO_PERMISSION	This error occurs when a security mechanism that is specified in the CSv2 token is not implemented on the server.	No	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	No
CredentialTokenNotSet	49421095	NO_PERMISSION	This is used to indicate that the token inside the credential is null.	No	No
InvalidEvidence	49421096	NO_PERMISSION	This error indicates that client authentication is required at the server. However, authentication information is not present in the method request from the client.	No	No

Table 12. CORBA minor codes after running a security-related request such as authentication (continued). The following table shows the CORBA minor codes which a client can expect to receive after running a security-related request such as authentication. The client can be either a stand-alone client or a server acting as a client. It also includes the CORBA exception type that the minor code would appear in.

Minor code name	Minor code value (in hex)	Exception type (all in the package of org.omg.CORBA .*)	Minor code description	Retry performed by stand-alone client (when authenticationRetryEnabled = true)	Retry performed
UserRegistryMethod_Protected	49421098	NO_PERMISSION	This error indicates that an attempt was made to remotely access a protected UserRegistry method.	No	No
ServerConnectionFailed	494210A0	COMM_FAILURE	This error is used when a connection attempt fails.	Yes (via ORB retry)	Yes (via ORB retry)
CorbaSystemException	494210B0	INTERNAL	This code is a generic CORBA specific exception in system code.	No	No
JavaException	494210B1	INTERNAL	This is a generic error that indicated that an unexpected Java exception occurred.	No	No
ValueIsNull	494210B2	INTERNAL	This code is used to indicate that a value or parameter that passed in is null.	No	No
EffectivePolicyNotPresent	494210B3	INTERNAL	This indicates that an effective policy object for CSrv2 is not present. This object is used to determine what security configuration features are specified.	No	No
NullPointerException	494210B4	INTERNAL	This code is used to indicate that a NullPointerException is caught in the runtime.	No	No
ErrorGettingClassInstance	494210B5	INTERNAL	This indicates a problem loading a class dynamically.	No	No
MalFormedParameters	494210B6	INTERNAL	This indicates parameters are not valid.	No	No
DuplicateSecurityAttributeType	494210B7	INTERNAL	This indicates a duplicate credential attribute that is specified during the set_attributes operation.	No	No
MethodNotImplemented	494210C0	NO_IMPLEMENT	This indicates that a method invoked is not implemented.	No	No
GSSFormatError	494210C5	BAD_PARAM	This code indicates that a Generic Security Services (GSS) encoding or decoding routine has created an exception.	No	No
TagComponentFormatError	494210C6	BAD_PARAM	This code indicates that a tag component cannot be read properly.	No	No
InvalidSecurityAttributeType	494210C7	BAD_PARAM	This code indicates an attribute type specified during the set_attributes operation is not a valid type.	No	No
SecurityConfigError	494210CA	INITIALIZE	This code indicates a problem exists between the client and server configuration.	No	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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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 ”
- “The setupClient.bat or setupClient.sh file is not working correctly”
- “WebSphere Application Server Version 6 is not working correctly with Enterprise Workload Manager (EWLM)” on page 113
- 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 113
- “When administrative security is enabled but application security is not enabled, the performance servlet displays authorization errors and cannot provide statistics” on page 113
- ““Name value is invalid” displays when migrating users and groups after the JACC provider for Tivoli is configured” on page 114
- “A Sun JDK can not read a PKCS12 keystore created by the Application Server” on page 114
- “Calling a secure resource from a non-secure resource is not supported when the non-secure resource collects data from users and then posts this data to the secure resource” on page 114

“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 >LTPA** .
- Complete the password and confirm password fields.
- Click **OK**.
- Try setting administrative or application security again.

The setupClient.bat or setupClient.sh file is not working correctly

The setupClient.bat file on Windows operating systems and the setupClient.sh file on Linux and UNIX-based platforms incorrectly specify the location of the SOAP security properties file.

In the setupClient.sh file, the CLIENTSOAP variable is:

```
CLIENTSOAP=-Dcom.ibm.SOAP.ConfigURL=file:$WAS_HOME/properties/soap.client.props
```

In the setupClient.bat and the setupClient.sh files, complete the following steps:

1. Remove the leading slash (/) after file:.
2. Change sas to soap.

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 appropriate output of the job log file.

```
<specialSubjects> name value is invalid
```

Note: Because the SystemOut log file does not exist on the z/OS operating system, check the output of the appropriate job log on the z/OS operating system.

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 padadmin 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.

Calling a secure resource from a non-secure resource is not supported when the non-secure resource collects data from users and then posts this data to the secure resource

If you have a non-secure resource (such as a JSP or a servlet) that calls a secure resource, the application might fail if the non-secure resource collects data from users and then posts this data to secure JSP or servlet files for processing.

To avoid this situation, structure your web application so that users are forced to login before the application performs any HTTP POST actions to the secure JSP or servlet files. To accomplish this, secure the first resource using whatever security mechanism that you choose (such as basic auth, form-login or cert).

This restriction is because basic auth and form-login use the servlet sendRedirect method, which has several implications for the user. The sendRedirect method is used twice during basic auth and form-login.

The sendRedirect method initially displays the basic auth or form-login page in the web browser. It later redirects the web browser back to the originally requested protected page. The sendRedirect(String URL) method tells the web browser to use the HTTP GET request to access the page that is specified in the web address. If HTTP POST is the first request to a protected JSP or servlet file, and no previous authentication or login occurred, then HTTP POST is not delivered to the requested page. However, HTTP GET is delivered because basic auth and form-login use the sendRedirect method, which behaves as an HTTP GET request which attempts to display a requested page after a login occurs.

Security enablement followed by errors

Use this information if you are experiencing errors after security is enabled.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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 116
- 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
- Error Message: CWSCJ0336E: Authentication failed for user {0} because of the following exception {1}
- “Error Message: An unexpected exception occurred initializing the security collaborator.java.lang.SecurityException: AuthConfigFactory error: java.lang.ClassNotFoundException: org.apache.geronimo.components.jaspi.AuthConfigFactoryImpl” on page 119
- “Error Message: SECJ8032W: AuthConfigFactory is undefined, using the default JASPI factory implementation class” on page 119
- “Error Message: SECJ0352E: Could not get the users matching the pattern {0} because of the following exception {1}” on page 120
- “Generate keys error when using the Profile Management tool to create a new profile” on page 120
- “Some security roles are not immediately available for a secured application where LDAP has Tivoli Access Manager enabled” on page 121
- “ABEND WITH ABEND EC3 REASON=020F2001” on page 121

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/software/webservers/appserv/zos_os390/support/

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 stand-alone program, this program might not gather or send credential information correctly.

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.

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.

- 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
/WebSphere/V6R1M0/AppServer/systemApps/isclite.ear/isclite.war/WEB-INF/validation.xml read)
```

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
```

```
/WebSphere/V6R1M0/AppServer1/profiles/profile_name/config/cells/cell_name/nodes/node_name/app.policy.
```

Where:

- V6R1M0 represents the version of WebSphere Application Server that you are using.
- 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: CSCJ0325W: 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.

– 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 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.

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: An unexpected exception occurred initializing the security collaborator.java.lang.SecurityException: AuthConfigFactory error: java.lang.ClassNotFoundException: org.apache.geronimo.components.jaspi.AuthConfigFactoryImpl

This error message occurs when your java.security file is missing an entry for the JASPI Provider. The default location for the java.security file is install_dir/properties. Edit the java.security file and add the following lines to it.

```
#
# The fully qualified class name of the default JASPI factory implementation class.
#
authconfigprovider.factory=com.ibm.ws.security.jaspi.ProviderRegistry
```

Note: This error only appears if you explicitly set your configuration to use this class. Otherwise, you might see error message SECJ8032W below.

Error Message: SECJ8032W: AuthConfigFactory is undefined, using the default JASPI factory implementation class

This error message occurs if the JASPI factory implementation is not defined. The default JASPI factory implementation has been set in the server runtime. However, JASPI might not function for a client.

To resolve, set the fully qualified class name of the default JASPI factory implementation class as the value for the property `authconfigprovider.factory` in the `java.security` file as in below:

```
#  
# The fully qualified class name of the default JASPI factory implementation class.  
#  
authconfigprovider.factory=com.ibm.ws.security.jaspi.ProviderRegistry
```

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.

CAUTION: 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.
Attention: 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 stand-alone 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

Some security roles are not immediately available for a secured application where LDAP has Tivoli Access Manager enabled

In some instances, some security roles might not be immediately available when you deploy a secured application where LDAP has Tivoli Access Manager enabled.

You might see an error such as the following:

```
"Exception: java.lang.OutOfMemoryError"
```

You might be able to address this issue by doing the following:

1. Allocate more memory to the minimum and maximum java heap size.
 - a. In the administrative console, select **Servers/server types/WebSphere Application servers --> server1**.
 - b. Select **Server Infrastructure/Java and Process Management/Process Definition**.
 - c. Select **Java Virtual Machine**.
 - d. Set the initial heap size to 512 MB and the maximum heap size to 1024 MB.
 - e. Select **OK** and then **Save**.
 - f. Restart WebSphere Application Server.
2. While WebSphere Application Server is stopped, add some performance tuning properties for embedded Tivoli Access Manager.
 - a. In the `config/cells/CELLNAME` directory, edit the `amwas.amjacc.template.properties` file by adding the following properties to it:

```
com.tivoli.pd.as.jacc.DBRefresh=0  
com.tivoli.pd.as.jacc.AuthTableRemoteMode=yes  
com.tivoli.pd.as.rbpf.NoUncheckedRoles=true
```

This helps when embedded Tivoli Access Manager is re-configured

- b. Because embedded Tivoli Access Manager is already configured, you can update the generated configuration files with the above properties. For each WebSphere Application Server instance in the ND (dmgr, NAs and servers), go to the `profiles/NAME/etc/tam` directory and do the following.

For each file that ends in `amjacc.properties`, add the 3 properties above:

```
com.tivoli.pd.as.jacc.DBRefresh=0  
com.tivoli.pd.as.jacc.AuthTableRemoteMode=yes  
com.tivoli.pd.as.rbpf.NoUncheckedRoles=true
```

For each file that ends in `pdperm.properties`, update the `appsvr-dbrefresh` property to be:

```
appsvr-dbrefresh=0
```

For each file that ends in `authztable.pdperm.properties`, update the `appsvr-mode` property to be:

```
appsvr-mode=remote
```

3. Restart the cell.

ABEND WITH ABEND EC3 REASON=020F2001

If security is not enabled either with zPMT dialogs or with ISPF customization dialogs immediately at installation time of the WebSphere Application Server for z/OS, the RACF definitions will not have been

completely generated. When security is enabled later using the administrative console, a missing RACF statement prevents the WebSphere Application Server control region from starting. Review APAR PK36598 for more details on resolving this problem.

Access problems after enabling security

Use this information if you are experiencing access problems after enabling security.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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”
- “I cannot access a web page after enabling security” on page 123
- “Authentication error accessing a Web page” on page 115
- “Authorization error accessing a Web page” on page 116
- “The client cannot access an enterprise bean after enabling security” on page 123
- “Client program never gets prompted when accessing secured enterprise bean” on page 124
- “Cannot stop an application server, node manager, or node after enabling security” on page 124
- “After enabling single sign-on, I cannot logon to the administrative console” on page 124
- “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 125
- “A Name NotFoundException error occurs when initially connecting to the federated repositories.” on page 125

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 logs of the application server which hosts the administrative console page for a related error message.

Note: You will need to use the administrative console to complete the next two items. If you are having a problem accessing the administrative console, you will have to turn off security and restart the administrative console. To turn off security, enter the following command at the system command prompt:

```
wsadmin.sh -conntype NONE
```

When the system command prompt displays again, enter:

```
securityoff
```

Restart the deployment manager after you turn off security.

- 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 trusted application functionality is enabled. The trusted application functionality is enabled if WebSphere Application Server has SAF access of READ to the RACF class of FACILITY, and profile of BBO.TRUSTEDAPPS.<cell short name>.<cluster short name>.

Remember: You could encounter synchronization problems if you are in a WebSphere Application Server, Network Deployment environment and save your security settings and the node agent was down.

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 logs for errors that relate 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 you are using Local OS and System Authorization Facility (SAF) Authorization, check the SAF EJBROLES setup. Specifically, verify that
 - the EJBROLE class is activated.
 - The role is defined to SAF.
 - The userid is permitted to the role.
 - The class is refreshed after the permit operation.

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.

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.

After enabling single sign-on, I cannot login 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 login page instead of the administrative console.

To correct this problem, use the fully qualified host name of the server, for example `http://myserver.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 NameNotFoundException 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 output of the appropriate job log:

```
NMSV0612W: A NameNotFound Exception
```

Note: Because the SystemOut.log file does not exist on the z/OS operating system, check the output of the appropriate job log on the z/OS operating system.

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.xml file. You can ignore this warning message. The message does not display when the wimDS database repository is configured.

Note: For IBM extension and binding files, the .xml or .xmi file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named ibm-*-ext.xml or ibm-*-bnd.xml where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The ibm-webservices-ext.xml, ibm-webservices-bnd.xml, ibm-webservicesclient-bnd.xml, ibm-webservicesclient-ext.xml, and ibm-portlet-ext.xml files continue to use the .xmi file extensions.

SSL errors for security

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" on page 126
- "Accessing resources using HTTPS" on page 126

- “javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure” on page 127
- “javax.net.ssl.SSLHandshakeException: unknown certificate” on page 127
- “javax.net.ssl.SSLHandshakeException: bad certificate” on page 128
-
- ““Catalog” tablet is blank (no item displayed) in GUI application client” on page 128
- “Modifying SSL Configurations after migration using -scriptCompatibility true” on page 128
- “Stand-Alone configuration fails when digital certificates are defined with the NOTRUST option” on page 129

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:

```
CWMMU0509I: The server "nodeagent" cannot be reached. It appears to be stopped.
CWMMU0211I: 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.

To end the processes, use the MVS™ console and type `c process_name`.

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:
`install_root/logs/server1.pid`

Accessing resources using HTTPS

If you are unable to access resources using a Secure Sockets Layer (SSL) URL (beginning with https:), or encounter error messages that indicate SSL problems, verify that your HTTP server is configured correctly for SSL. Browse the welcome page of the HTTP server using SSL by entering the URL:
`https://host_name`.

If the page works with HTTP, but not HTTPS, the problem is with the HTTP server.

- Refer to the documentation for your HTTP server for instructions on correctly enabling SSL. If you are using the IBM HTTP Server or Apache, go to: <http://www.ibm.com/software/webservers/httppervers/library.html>. Click **Frequently Asked Questions > SSL**.
- If you use the IBM Key Management (IKeyman) tool to create certificates and keys, remember to stash the password to a file when creating the Key Database (KDB) file with the IBM Key Management Tool.
 1. Go to the directory where the KDB file is created, and see if an .sth file exists.
 2. If not, open the KDB file with the IBM Key Management Tool, and click **Key Database File > Stash Password**. The following message is displayed: The password has been encrypted and saved in the file.

If the HTTP server handles SSL-encrypted requests successfully, or is not involved (for example, traffic flows from a Java client application directly to an enterprise bean that is hosted by WebSphere Application Server, or the problem displays only after enabling WebSphere Application Server security), what kind of error are you seeing?

System SSL: See *z/OS System Secure Sockets Layer Programming SC24-5901* for information on using the System Secure Sockets Layer (SSL) callable services programming interfaces.

- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure
- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: unknown certificate
- javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: bad certificate

For general tips on diagnosing and resolving security-related problems, see “Security components troubleshooting tips” on page 99

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

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.TRANSPORT_EXCEPTION: 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 *install_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.

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.TRANSPORT_EXCEPTION: 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: CWMJE0080E: 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.

"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 truststore 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 Version 6.1 and later releases. New capabilities were added that are not supported when

the configurations are not migrated to the latest format. If you are migrating from a release prior to Version 6.1, you can use the `convertSSLConfig` task to convert your SSL configuration information to the centralized SSL configuration format.

Stand-Alone configuration fails when digital certificates are defined with the NOTRUST option

If your digital certificates are defined with the NOTRUST option, it is possible that you might receive the following error message:

```
Trace: 2008/06/18 16:57:57.798 01 t=8C50B8 c=UNK key=S2 (0000000A)
Description: Log Boss/390 Error
from filename: ./bbgcfcom.cpp
at line: 376
error message: BB000042E Function AsyncIOaccept failed with RV=-1, RC=124, RSN=050B0146, ?EDC5124I
Too many open files. (errno2=0x0594003D)??
```

If this error appears, enter 'D 0MVS,P. If you have a NOTRUST issue a large number appears under 'OPNSOCK'.

Check your digital certificates and make sure they are not marked with the NOTRUST option. This can occur if the certificates were created with a date beyond the expiration date of the CERTAUTH that was used to create it.

Single sign-on configuration troubleshooting tips for security

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:

```
% 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 website 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 websites 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:
 - a. Edit the Server document.
 - b. 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.

Unexpected re-authentications: When you set the `com.ibm.ws.security.web.logoutOnHTTPSessionExpire` custom property to `true`, unexpected re-authentications might occur when you are working with multiple web applications. By default, each web application has its own unique HTTP session, but the web browser has one session cookie. To address this issue, you can change the HTTP session configuration by giving each application a unique session cookie name or path setting. As a result, each application gets its own session cookie. Alternatively, you can configure multiple web applications with the same enterprise application to share the same HTTP session. For more information, see the Assembling so that session data can be shared topic.

- Possible issues when SSO is enabled and Firefox v3.6.11 is configured to accept third-party cookies. If you have SSO enabled, and when using Firefox v3.6.11 one of the following is true:
 - It is configured to accept third-party cookies that are kept until they expire or until Firefox is closed
 - You have one session open but are switching to different applications
 - More than one session is opened for different applications that require different users for authorization

you might see the following error message: `Error 403: AuthorizationFailed`.

To resolve, clear the third-party cookies before launching a new application by doing the following:

1. Select **Firefox Tools > Options > Privacy**.
2. Ensure that the history is set to **Remember History**.
3. Click on **Remove individual cookies** to delete the cookies.

You can also close other sessions if Firefox is configured to accept third-party cookies that are kept until Firefox is closed.

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

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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 135
- "An "HPDBA0219E: An error occurred reading data from an SSL connection" might occur" on page 136
- A "There are no ports available in the port set" error might occur

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.commonconfig.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 WebSphere Application Server, Network Deployment (ND) environment with N servers defined (for example, server1, server2).

The following files must be removed.

```
install_root/tivoli/tam/PdPerm.properties
install_root/tivoli/tam/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:

```
install_root/java/jre/bin/java -classpath
"install_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, 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.

A "There are no ports available in the port set" error might occur

When you use Tivoli Access Manager as the JACC provider and stop WebSphere Application Server using the administrative console or the wsadmin script, there is a clean up process that runs for Tivoli Access Manager. WebSphere Application Server is unable to complete the clean up process.

WebSphere Application Server uses a different port number for each new process. Eventually, the application server runs out of port numbers to connect to the Tivoli Access Manager server and displays a "There are no ports available in the port set" error.

If this error occurs, you must manually clean up the ports that are available to WebSphere Application Server processes. A script is provided to complete this process. For more information, see Cancelling multiple processes with Tivoli Access Manager as the Java Authorization Contract for Containers (JACC) provider might cause a "There are no ports available in the port set" error .

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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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 13. SPNEGO TAI trace specifications.

This table describes the SPNEGO TAI trace specifications.

Trace	Use
<code>com.ibm.security.jgss.debug</code>	Set this JVM Custom Property to <code>all</code> to trace through JGSS code. Messages appear in the <code>trace.log</code> file, and SystemOut.log .
<code>com.ibm.security.krb5.Krb5Debug</code>	Set this JVM Custom Property to <code>all</code> to trace through the Kerberos5-specific JGSS code. Messages appear in the <code>trace.log</code> file, and SystemOut.log .
<code>com.ibm.ws.security.spnego.*</code>	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 <code>trace.log</code> 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 : C1 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 0 [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 ServerCredential >
com.ibm.ws.security.spnego.ServerCredential initialize ENTRY
SPNEG0014: Kerberos initialization Failure: org.ietf.jgss.GSSEException, 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.throwGSSEException
(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.GSSEException, 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: <pre>[2/27/06 14:28:04:191 CST] 00000059 SpnegoHandler < com.ibm.ws.security.spnego.SpnegoHandler handleRequest: Received a non-SPNEGO Authorization Header RETURN</pre>
Description	<p>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:</p> <ul style="list-style-type: none">• 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 <code>http://localhostsomeURL</code> instead of a fully qualified name.• The SPN is not found in the Active Directory. The SPN must be of the format <code>HTTP/server.realm.com</code>. The command to add the SPN is <pre>setspn -a HTTP/server.realm.com userid</pre>• 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<ul style="list-style-type: none">– <code>setspn -a HTTP/myappserver.austin.ibm.com user1</code>– <code>setspn -a HTTP/myappserver.austin.ibm.com user2</code>Same SPN and same user IDs, one with a port number defined<ul style="list-style-type: none">– <code>setspn -a HTTP/myappserver.austin.ibm.com user3</code>– <code>setspn -a HTTP/myappserver.austin.ibm.com:9080 user3</code>
User Action	<p>If the SPN is defined incorrectly as <code>HTTP/server.realm.com@REALM.COM</code> with the addition of <code>@REALM.COM</code>, then delete the user, redefine the user, and redefine the SPN.</p> <p>If the problem is with the Kerberos keytab file, then regenerate the keytab file.</p> <p>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:</p> <pre>setspn ?L userid</pre> <p>Returns the message, cannot find account userid, if the SPN is not registered.</p> <pre>setspn -L</pre> <p>Displays the SPNs that exist.</p>

Problem: Credential Delegation is not working.

Symptom	An invalid option is detected. When tracing is enabled, the following message is displayed: <pre>com.ibm.security.krb5.KrbException, status code: 101 message: Invalid option in ticket request</pre>
Description	The Kerberos configuration file is not properly configured.
User Action	Ensure that neither <code>renewable</code> , nor <code>proxiable</code> are set to <code>true</code> .

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: <pre>com.ibm.security.krb5.internal.crypto.KrbCryptoException, status code: 0 message: Checksum error; received checksum does not match computed checksum</pre>
----------------	---

Description RC4-HMAC encryption is not supported with a Microsoft Windows version prior to 2003 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 version prior to 2003 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 version prior to 2003 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.

Description Authorization: Negotiate YII.....

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.
Description	<p>"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.</p> <p>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.</p> <p>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.</p>

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 or 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 *****
<code>com.ibm.ws.security.spnego.SPN1.hostName</code>	<code>server.wastedhed30.torolab.ibm.com</code>
<code>com.ibm.ws.security.spnego.SPN1.filterClass</code>	<code>com.ibm.ws.security.spnego.HTTPHeaderFilter</code>
<code>com.ibm.ws.security.spnego.SPN1.filter</code>	<code>request-ur!=noSPNEGO</code>
<code>com.ibm.ws.security.spnego.SPN1.NTLMTokenReceivedPage</code>	<code>File:///C:/temp/NTLM.html</code>

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>
```

Problem: The trust association interceptor (TAI) does not call the initialize(Properties) method

Tracing might show that TAI is loaded, but that the `initialize(Properties)` method is not called. Only the `getVersion()` method appears to be called during startup.

WebSphere's TAI processing only calls `initialize(Properties)` when there are custom properties defined for the TAI.

To fix this issue, define an unused TAI custom property, such as `com.ibm.iissw.spnegoTAI.NumberOfServers=0`.

Problem: The trust association interceptor (TAI) is not loading properly

Tracing might show that TAI is not loading and the following exception text is received:

```
SPNEGO014: Kerberos initialization Failure: org.ietf.jgss.GSSEException, major code: 13, minor code: 0
  major string: Invalid credentials
  minor string: SubjectKeyFinder: no JAAS Subject
  at com.ibm.security.jgss.i18n.I18NException.throwGSSEException(I18NException.java:12)
  ...
```

A possible cause for this is that the JVM custom property `javax.security.auth.useSubjectCredsOnly` is not set to a value of `false`.

To fix this issue, define a JVM custom property on each JVM that is enabled for the TAI, `javax.security.auth.useSubjectCredsOnly=false`.

Problem: JACL scripts default characters for adding trust association interceptor (TAI) parameters can cause issues

JACL scripts for adding TAI parameters accept positional parameters. To accept the defaults, a `“.”` is specified. On some WebSphere platforms, if you specify a `“.”` it can cause the property to be added with a value of `“.”`.

Always (regardless of platform), confirm that the properties added are as expected using the administrative console. If they are not, manually correct them.

SPNEGO troubleshooting tips

You can securely negotiate and authenticate HTTP requests for secured resources in WebSphere Application Server by using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO). This article describes the issues you might encounter using Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) as the web authentication service for WebSphere Application Server.

SPNEGO issues and their possible solutions

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

The following are some issues you might encounter when you use SPNEGO as the web authentication service for WebSphere Application Server and their possible solutions.

- “Unable to resolve the Kerberos principal name” on page 144
- “WebSphere Application Server and the time on the Active Directory (AD) domain controller are not synchronized within 5 minutes” on page 144
- “No factory is available to create name for mechanism 1.3.6.1.5.5.2” on page 145
- “A Kerberos error is received while decoding and verifying the SPNEGO token” on page 145
- “Single sign-on does not occur” on page 146
- “Unable to use sign-on (SSO) with RC4-HMAC encryption” on page 146
- “Problems when accessing a protected URL through the SPNEGO single sign-on (SSO)” on page 147
- “Even with JGSS tracing disabled, some KRB_DBG_KDC messages appear in the SystemOut.log” on page 148
- “ktpass is unable to find the userid” on page 148

- “Credential delegation might not work due to an invalid option in the ticket request” on page 147
- “A user is challenged for credentials even though the browser is properly configured” on page 149
- “A user using the Novell client cannot authenticate using SPNEGO” on page 149
- “Accessing SPNEGO sites via some caching proxy servers can cause SPNEGO authentication issues” on page 149
- “Virtual Private Networks (VPN) software and firewalls might interfere with SPNEGO operations” on page 149
- “Possible browser issue when accessing a SPNEGO protected application” on page 150
- “Possible browser issue with Internet Explorer 6.0” on page 150
- “Error pages defined for the NTLMTokenReceivedPage or the SpnegoNotSupportedPage properties do load from an http:// URL” on page 150
- “A client browser single sign-on (SSO) attempt fails to authenticate with WebSphere Application Server when you use a SPNEGO token with Microsoft Internet Security Acceleration Server” on page 150
- “Microsoft Windows Version 7 and Internet Explorer Version 8 disables DES encryption type by default” on page 151

Unable to resolve the Kerberos principal name

If you are unable to resolve the Kerberos principal name, as shown in the following trace example:

```
[11/11/03 1:42:29:795 EST] 1d01b21e GetKrbToken > Negotiation (GSS): Begin handshake
[11/11/03 1:42:29:795 EST] 1d01b21e Context > GSS Context init, servername:HTTP@johnwang5.jwcmd.com
[11/11/03 1:42:29:866 EST] 1d01b21e TraceNLS u No message text associated with key Error.getting.the.Token,
.GSS.Exception:org.ietf.jgss.GSSException, .major.code:.13, .minor.code:.0
major.string:.Invalid.credentials
minor.string:.Cannot.get.credential.from.JAAS.Subject.for.principal:.HTTP/192.168.0.4@168.0.4 in bundle
com.ibm.ejs.resources.security
[11/11/03 1:42:29:866 EST] 1d01b21e GetKrbToken E Error getting the Token, GSS Exception:org.ietf.jgss.GSSException,
major code: 13, minor code: 0
major string: Invalid credentials
minor string: Cannot get credential from JAAS Subject for principal: HTTP/192.168.0.4@168.0.4
[11/11/03 1:42:29:876 EST] 1d01b21e TraceNLS u No message text associated with key SpnegoTAI.exits.due.to.an.exception.
in bundle com.ibm.ejs.resources.security
[11/11/03 1:42:29:876 EST] 1d01b21e SpnegoTAI E SpnegoTAI exits due to an exception.
```

add the IP address of the server in its host file. You must also recycle the application server to load the new host file.

WebSphere Application Server and the time on the Active Directory (AD) domain controller are not synchronized within 5 minutes

The trace.log file for this issue is similar to the following:

```
[11/11/03 1:44:09:499 EST] 1d01b21e GetKrbToken > Negotiation (GSS): Begin handshake
[11/11/03 1:44:09:499 EST] 1d01b21e Context > GSS Context init, servername:HTTP@backendrc4.ibm.net
[11/11/03 1:44:09:499 EST] 1d01b21e Context > GSS Context init, done.
[11/11/03 1:44:09:679 EST] 1d01b21e SpnegoTAI > Server response token as follows...
0000: 6082014f 06062b06 01050502 a1820143 ~?.0..+.....i?.C
0010: 3082013f a0030a01 01a10b06 092a8648 0?.? ?...i...*?H
0020: 82f71201 0202a282 01290482 01256082 ?+....{?.)?..?`?
0030: 01210609 2a864886 f7120102 0203007e !!..*?H?+.....~
0040: 82011030 82010ca0 03020105 a1030201 ?..0?.. ....i...
0050: 1ea41118 0f323030 33313131 31303634 .R...20031111064
0060: 3430395a a5050203 0a3548a6 03020125 409Z%....5H!...%
0070: a90b1b09 4a57434d 442e434f 4daa2630 @....IBM.NET#&0
0080: 24a00302 0100a11d 301b1b04 48545450 $ ....i.0...HTTP
0090: 1b136a6f 686e7761 6e67352e 6a77636d ..backendrc4.ibm
00a0: 642e636f 6dab81ab 1b81a86f 72672e69 .net.«?«.?? org.i
00b0: 6574662e 6a677373 2e475353 45786365 etf.jgss.GSSExce
00c0: 7074696f 6e2c206d 616a6f72 20636f64 ption, major cod
00d0: 653a2031 302c206d 696e6f72 20636f64 e: 10, minor cod
00e0: 653a2033 370a096d 616a6f72 20737472 e: 37..major str
00f0: 696e673a 20446566 65637469 76652074 ing: Defective t
0100: 6f6b656e 0a096d69 6e6f7220 73747269 oken..minor stri
0110: 6e673a20 436c6965 6e742074 696d6520 ng: Client time
0120: 54756573 6461792c 204e6f76 656d6265 Tuesday, Novembe
0130: 72203131 2c203230 30332061 7420313a r 11, 2003 at 1:
0140: 33353a30 3120414d 20746f6f 20736b65 35:01 AM too ske
0150: 776564 wed
```

You can fix this issue in one of two ways. The preferred method is to synchronize the WebSphere system time to within 5 minutes of the time of the AD server. A best practice is to use a time server to keep all of the systems synchronized. Alternatively, you can also add or adjust the clockskew parameter in the Kerberos configuration file. Note that the default is 300 seconds (5 minutes).

No factory is available to create name for mechanism 1.3.6.1.5.5.2

If the **systemout.log** file contains an exception error similar to the following:

```
[4/8/05 22:51:24:542 EDT] 5003e481 SystemOut    0 [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 E com.ibm.issw.spnegoTAI.ServerCredential initialize() SPNEG0014:
Kerberos initialization Failure: org.ietf.jgss.GSSEException, 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.throwGSSEException(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)
.
```

make sure that the **java.security** file contains the IBMSPNego security provider and is defined correctly. It should contain a line similar to the following:

```
security.provider.6=com.ibm.security.jgss.mech.spnego.IBMSPNego
```

A Kerberos error is received while decoding and verifying the SPNEGO token

You might receive the following exception error as the Java Generic Security Service (JGSS) library attempts to process the SPNEGO token:

```
Error authenticating request. Reporting to client
Major code = 11, Minor code = 31
org.ietf.jgss.GSSEException, 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
```

This error is caused when the ticket is encoded by using one key and then an attempt is made to decode the ticket by using another key. There are number of possible explanations for this:

- The keytab file has not been copied to the server machine after it has been regenerated.
- The Kerberos configuration points to the wrong keytab file.
- The SPN was defined to Active Directory more than once. This is also caused by another userid with a similarly defined SPN (either the same name or it might differ by having a port defined as part of the SPN).
- If the encryption type is DES, the password associated with the Service userid might only exist for RC4-HMAC encryption. This occurs when a new userid is created, the SPN is defined, and the keytab is generated with the +Des0nly option. The service ticket generated for this SPN is encrypted with one secret that does not match that found in the keytab.
- An older version of the Microsoft ktpass tool is being used. Older versions of the tool create keytab files that are incorrect and might result in this error. If you are using Windows Server 2003 as your Domain controller, use the version of **ktpass.exe** that is part of Windows Server 2003 SP 2 (specifically, version 5.2.3790.2825).

If the problem is with the keytab file, then fix it. If the problem is with multiple SPN definitions, remove the extra or conflicting SPN, confirm that the SPN is no longer registered with AD, and then add the SPN again. Read about Creating a Kerberos service principal name and keytab file for more information. The Active Directory might need to be searched for other entries with SPNs defined that clash with the SPN using an LDAP browser.

To confirm that the SPN is not registered, the following command:

```
setspn -l userid
```

should return with:

```
Cannot find account userid
```

If the userid and keytab are for DES-CBC-MD5, after you create the userid, change the password for the userid and then create the keytab file. If you are using Windows Server 2003 upgrade to the latest version of ktpass.

Single sign-on does not occur

When trace is turned on, the following error message might appear:

```
Client sent back a non-SPNEGO authentication header, SpnegoTAI exits
```

A possible reason for this error is that the client is returning an NT LAN manager (NTLM) response to the authorize challenge, not an SPNEGO token. This can occur due to one or more of the following issues:

- The client has not been properly configured.
- The client is not using a supported browser. For instance, users of Internet Explorer 5.5 SP1 respond with a non-SPNEGO authentication header.
- The user has not logged into the AD domain or into a trusted domain, or the client used does not support Integrated Authentication with Windows. In this case, the TAI is working properly.
- The user accesses a service defined on the same machine as the client is running (the localhost). Internet Explorer resolves the hostname of the URL to `http://localhost<someURL>` instead of to the fully-qualified name that is provided.
- 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
```

If the SPN is defined incorrectly as `HTTP/server.realm.com@REALM.COM` with the addition of `@REALM.COM`, then delete the user, redefine it, and then redefine the SPN.

- The hostname is resolved as a DNS Alias, not as a HOST record. Change the hostname to a HOST record.
- The account in AD that holds the ServicePrincipalName is in an AD domain that is remote from the AD domain that the user has logged into, and these domains are not Windows 2003 domains. Migrate the domains to Windows 2003, or limit SSO to users within the same domain as the ServicePrincipalName userid.

Unable to use sign-on (SSO) with RC4-HMAC encryption

When trace is turned on you might receive the following error message:

```
com.ibm.security.krb5.internal.crypto.KrbCryptoException, status code: 0  
message: Checksum error; received checksum does not match computed checksum
```

Some possible reasons for this error include the following

- RC4-HMAC encryption is not supported with a Windows version prior to 2003 KDC. To confirm that this is a problem, examine the trace above where the exception is thrown. The content of the incoming ticket should be visible in the trace. While it is encrypted, the SPN name for the service is readable. If a Windows version prior to 2003 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 shown. For example, this is beginning of the ticket received from a Windows version prior to 2003 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..

```

To correct the problem, either use single DES encryption or use a Windows Server 2003 for a KDC. Remember to regenerate the SPN and the keytab file.

- RC-HMAC encryption does not work when the credential delegation feature is used. To determine if you have this problem, enable JGSS and Krb5 tracing. If the SPN name is correct, messages such as the following might appear:

```

[JGSS_DBG_CTX] Successfully decrypted ticket
[JGSS_DBG_CTX] Put authz info in cache
[JGSS_DBG_CTX] Session key type = rc4-hmac
...
[JGSS_DBG_CTX] Successfully decrypted authenticator
[JGSS_DBG_CTX] Error authenticating request. Reporting to client
...
Major code = 11, Minor code = 0
org.ietf.jgss.GSSException, major code: 11, minor code: 0
major string: General failure, unspecified at GSSAPI level
minor string: Kerberos error converting KRBCred: com.ibm.security.krb5.internal.crypto.KrbCryptoException, status code: 0
message: Checksum error; received checksum does not match computed checksum

```

This indicates that the delegated credential contained in the SPNEGO token was not encrypted with the proper key.

Obtain APAR IY76826. This replaces **ibmjgssprovider.jar** with a version that can accept the Microsoft defined RC4 encrypted delegated credential.

- The password used when generating the keytab file with ktpass does not match the password assigned to the service account. When the password changes you should regenerate and redistribute the keys., even if it is reset to the same password.

In addition, the ktpass tool might generate a keytab file with a non-matching password as in the following cases:

- If the password entered to ktpass matches the password for the service account, then the produced keytab file does work.
- If the password entered to ktpass does not match the password for the service account, and is less than 7 characters in length, ktpass stops and does not produce a keytab file.
- If the password entered to ktpass does not match the password for the service account, and is greater than 6 characters in length, ktpass does not stop. Instead, it produces a keytab file containing an invalid key. Use of this key to decrypt a SPNEGO token produces the checksum error listed above.

Use a non-null password for the service account, and then use that password when invoking ktpass.

- The ktpass version 1830 (in Support Tools SP1) can produce the error in some Windows 2003 Server environments. Use the SP2 version of the tool to avoid the error.

Use the Support Tools SP2 version of ktpass to generate the keytab file.

Credential delegation might not work due to an invalid option in the ticket request

When trace is turned on, if the following error message appears:

```
com.ibm.security.krb5.KrbException, status code: 101 message: Invalid option in ticket request
```

the Kerberos configuration file is not properly configured. Ensure that neither renewable nor proxiable are set to true.

Problems when accessing a protected URL through the SPNEGO single sign-on (SSO)

You might receive an error similar to the following when accessing a protected URL through the SPNEGO SSO:

Bad Request

Your browser sent a request that this server could not understand.
Size of request header field exceeds server limit.

Authorization: Negotiate YII.....

This message is generated by the Apache/IBM HTTP Server, and indicates that the authorization header that your browser has returned is too large. The long string that follows the word Negotiate is the SPNEGO token. This SPNEGO token is a wrapper of the Windows Kerberos token. Windows includes the PAC information of the user 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 SPNEGO becomes. IBM HTTP Server 2.0 (as well as Apache 2.0 and IBM HTTP Server 6.0) limit the size of any acceptable HTTP header to be 8K. In Windows domains with many groups, and with user membership in many groups, the size of the user's SPNEGO token can exceed the 8K limit.

If possible, reduce the number of security groups that 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.

After applying the fix you must specify the LimitRequestFieldSize parameter in the **httpd.conf** file to increase the size of allowable headers from the default of 8192.

Even with JGSS tracing disabled, some KRB_DBG_KDC messages appear in the SystemOut.log

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 default value of the `krb5` property is to emit some messages to `SystemOut.log`.

To remove all KRB_DBG_KDC messages from the `SystemOut.log`, set the JVM property to `-Dcom.ibm.security.krb5.Krb5Debug=none`.

ktpass is unable to find the userid

When using `ktpass`, you might receive an error message similar to the following:

```
DsCrackNames returned 0x2 in the name entry for server3  
Failed getting target domain for specified user.
```

In an Active Directory forest, the `userid` lookup used by the `ktpass.exe` does not have a default domain name to be used. This does not occur when the domain controller is not in a forest.

To fix this problem, instead of specifying option `-mapUser userid`, use `-mapUser userid@domain` instead. For example, specify `-mapUser server3@WIBM.NET`.

Credential delegation does not work for any userid

If in the `trace.log`, an error exception similar to the following appears:

```
> com.ibm.issw.spnegoTAI.Context getDelegateCred() Entry  
d com.ibm.issw.spnegoTAI.Context getDelegateCred() unable to get Delegate Credential  
< com.ibm.issw.spnegoTAI.Context getDelegateCred() Exit  
W com.ibm.issw.spnegoTAI.SpnegoHandler handleRequest() SPNEG0021: No delegated credentials were found for user: nouser@NA.IBM.NET
```

the domain account on which the SPN is attached does not have the "Account is trusted for Delegation" property defined.

To address this issue, ensure that the domain account does define the "Account is trusted for Delegation" property.

A user is challenged for credentials even though the browser is properly configured

A user might be challenged for credentials even though the browser is configured properly. The TAI might have obtained the user's credentials from the SPNEGO token, and the user might have failed to log in. In the trace.log an exception error similar to the following appears:

```
< com.ibm.issw.spnegoTAI.SpnegoTAI getAuthenticatedUsername(): lansche Exit  
d com.ibm.issw.spnegoTAI.SpnegoTAI negotiateValidateandEstablishTrust(): Handshake finished, sending 200 :SC_OK  
< com.ibm.issw.spnegoTAI.SpnegoTAI negotiateAndValidateEstablishedTrust Exit  
A SECJ0222E: An unexpected exception occurred when trying to create a LoginContext. The LoginModule alias is system.WEB_INBOUND  
and the exception is...
```

The userid (which is lansche in the example above) does not exist in the registry in use by WebSphere. This problem can be caused when:

- The registry used by WebSphere is not the Active Directory domain LDAP, or Global Catalogue, but is some other virtual registry (for example, a file-based custom user registry).
- A custom IClientToServerUserIdMapper implementation modifies the username such that the name it is mapped to does not exist in the registry.
- The attribute mapped to by the WebSphere LDAP User Filter property is incorrect.

To fix this problem, ensure that the user that is being asserted to WebSphere Application Server by the TAI is the configured WebSphere registry.

A user using the Novell client cannot authenticate using SPNEGO

If a user using the Novell client cannot authenticate using SPNEGO they might receive a “An NTLM token is received.” message.

The user might have logged into the Novell Client but did not perform a Windows Kerberos login (this can be confirmed using the Kerbray utility). If a user has logged onto the Windows domain and has a Kerberos ticket, the user cannot utilize SPNEGO authentication.

To fix this problem, remove the Novell client and use the default Windows domain login.

Accessing SPNEGO sites via some caching proxy servers can cause SPNEGO authentication issues

If you access SPNEGO sites via some caching proxy servers you might not be able to authenticate using SPNEGO. The message “SPNEGO authentication not supported on this client” might be displayed.

It is possible that the caching proxy is changing the hostname that returns on the HTTP 401 Authenticate Negotiate response.

If you have this issue, contact your proxy vendor for a possible solution.

Virtual Private Networks (VPN) software and firewalls might interfere with SPNEGO operations

You might experience problems with VPN software and firewalls that might interfere with SPNEGO operations.

To resolve these issues, contact your VPN and or firewall vendors for any configuration changes that might be necessary.

Possible browser issue when accessing a SPNEGO protected application

There might be a browser issue if you log onto a domain machine using one password (for example, *passwordA*) and then log onto a second domain machine by changing your original password (for example, you might change your password on the second domain machine to *passwordB*).

Once you return to the original domain machine, you might not be able to obtain either a SPNEGO/Kerberos or an NTLM response to the Negotiate challenge. After two attempts, the browser displays an HTTP 404 error message.

To resolve this issue, log off the original domain machine and log back on with the new password (*passwordB*).

Possible browser issue with Internet Explorer 6.0

When WebSphere Application Server is configured with SPNEGO and fallback is enabled for a request, Internet Explorer 6.0 might fail to login to the form login pages.

To avoid this situation, complete one of the following actions:

- From the **Global security > SPNEGO Web Authentication** panel, deselect the **Allow fall back to application authentication mechanism** option if it is selected.
- Upgrade to Internet Explorer Version 7.0
- Configure Internet Explorer Version 6.0 to use a different authentication page. The issue is with the basic authentication versus the form login authentication preference.

Error pages defined for the NTLMTokenReceivedPage or the SpnegoNotSupportedPage properties do load from an http:// URL

The error pages defined for the NTLMTokenReceivedPage or the SpnegoNotSupportedPage properties do load from an http:// URL. The following trace message might appear:

```
Could not load the SPNEGO not supported content, going with the default content.  
Exception received: java.net.ProtocolException: Server redirected too many times (20)
```

This issue occurs when the loaded file performs an automatic redirect. It is not possible to both load the file from a web server and also use an automatic redirection

To resolve this issue, load the content from a file:/// URL, not an http:// URL.

A client browser single sign-on (SSO) attempt fails to authenticate with WebSphere Application Server when you use a SPNEGO token with Microsoft Internet Security Acceleration Server

When tracing is enabled, the following messages exist:

```
com.ibm.ws.security.spnego.SpnegoHandler isAuthHeaderNotSPNEGO  
ENTRY Negotiate
```

```
com.ibm.ws.security.spnego.SpnegoHandler isAuthHeaderNotSPNEGO  
Client sent back a non-SPNEGO authentication header
```

When a Microsoft Internet Security Acceleration Server (ISA) exists between a client browser and WebSphere Application Server, ISA might intercept the SPNEGO authentication header from the client browser request. ISA converts the SPNEGO object identifier (OID) to a Kerberos OID. The authentication attempt with WebSphere Application Server fails because the SPNEGO OID has been converted and is now missing.

For information about how to fix this issue, see the "Users cannot access a web site that is published in ISA Server 2006 if the web site accepts only the SPNEGO authentication package" topic on the Microsoft Corporation Support site.

Microsoft Windows Version 7 and Internet Explorer Version 8 disables DES encryption type by default

If you are using Microsoft Windows Version 7 with Internet Explorer Version 8, and you cannot get SPNEGO Single Sign On (SSO) to function, it could be because Windows Version 7 disabled DES encryption type for Kerberos by default. When trace is turned on the following message appears:

```
Client sent back a non-SPNEGO authentication header....
```

It is recommended that you change your encryption type to RC4-HMAC or to AES. If you still choose to use the DES encryption type, however, you must refer to the Windows 7 documentation for help on how to enable the DES encryption type.

The following is an example of how to change your encryption type from DES to RC4:

1. Make sure the Microsoft Active Directory account that you use to map to the SPN does not have the **Use DES encryption type for this account** box checked. In the Microsoft Active Directory machine:
 - a. Click **Start->Programs->Administrative Tools->Active Directory Users and Computers->Users**.
 - b. Click on the Microsoft Active Directory account that you use to map to the SPN.
 - c. Select the account, and then make sure that the **Use DES encryption type for this account** box is not checked.
2. Reset the password for the Microsoft Active Directory account that you use to map to the SPN. You can reset it to the same password.
3. Regenerate the keytab with the RC4 encryption type.
4. Copy the new keytab file to the WebSphere Application Server servers.
5. Update the Kerberos configuration (krb5.ini/krb5.conf) files to list RC4 first for the default_tkt_encetypes and default_tgs_encetypes attributes.

For example:

```
default_tkt_encetypes = rc4-hmac des-cbc-md5  
default_tgs_encetypes = rc4-hmac des-cbc-md5
```

6. Stop and restart all WebSphere Application Server servers.

Note: If you have more than one Microsoft Active Directory account that you use to map to different SPNs, then you must repeat steps 1 through 3 above for each SPN and the merging of all the keytab files.

Chapter 14. Troubleshooting Service integration

This page provides a starting point for finding information about service integration.

Service integration provides asynchronous messaging services. In asynchronous messaging, producing applications do not send messages directly to consuming applications. Instead, they send messages to destinations. Consuming applications receive messages from these destinations. A producing application can send a message and then continue processing without waiting until a consuming application receives the message. If necessary, the destination stores the message until the consuming application is ready to receive it.

Troubleshooting service integration technologies

Use this overview task to help resolve a problem that you think is related to service integration technologies.

Before you begin

For general information about fixing problems, see *Diagnosing problems (using diagnosis tools)*.

To help you identify and resolve problems, use the WebSphere Application Server tracing and logging facilities. For more information about using tracing and logging, see *Adding logging and tracing to your application*.

About this task

If you encounter a problem that you think might be related to service integration technologies, complete the following stages.

Procedure

1. Check the “Tips for troubleshooting” related links for specific problems related to service integration technologies.
2. Use “Troubleshooting service integration message problems” on page 164 to investigate problems with messages, such as messages not arriving or being consumed. If the tips and the investigations do not help you fix the problem, complete the following general stages.
3. Check the Release Notes for known problems and their resolution. The Release Notes are available from the WebSphere Application Server library website.
4. For current information from IBM Support on known problems and their resolution, see the WebSphere Application Server support page.
5. Search the WebSphere Technotes database.

You can use the following queries to find articles for service integration technologies:

- All technotes: <http://www.ibm.com/support/search.wss?rs=180tc=SSEQTP&tc1=SSCBRCs>
- MustGather technotes, used to debug problems: <http://www.ibm.com/support/search.wss?rs=180tc=SSEQTP&tc1=SSCBRCs&q=MustGather>

6. Check for error messages.

Check in the SYSPRINT or SYSOUT log for error messages.

The following message prefixes relate to specific aspects of service integration technologies:

Table 14. Prefixes and service integration technologies. The first column of the table lists the message prefixes. The second column contains the specific aspects of service integration technologies related to the message prefixes.

Message prefix	Aspect of service integration technologies
CWSIA	API
CWSIB	Message formatting and parsing core
CWSIC	Communications
CWSID	Administration and system management
CWSIE	Message formatting and parsing SPI
CWSIF	Message formatting and parsing
CWSIG	Example
CWSIH	Match space
CWSII	Security
CWSIJ	Communications formats and protocol
CWSIK	Common messages
CWSIL	Publish and subscribe bridge
CWSIM	Mediations
CWSIN	Mediation services
CWSIO	Administration migration
CWSIP	Message processor
CWSIQ	MQ formats and protocol
CWSIR	Core programming interface
CWSIS	Message store
CWSIT	Topology routing and management
CWSIU	Utilities
CWSIV	Resource adapter
CWSIX	Core beans
CWSIY	Mediation handler framework
CWSIZ	Mediation framework
CWSJA	Administration commands
CWSJB	Inter-bus link
CWSJC	Core selector
CWSJD	Administration security
CWSJO	Service Data Objects configuration
CWSJQ	Message formatting and parsing MQ interoperability
CWSJR	Resource adapter (JMS)
CWSJU	Message tracing
CWSJW	WLM classifier for z/OS

The Troubleshooter reference: Messages contains information about the 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.

7. Check for more information and error messages that might provide a clue to a related problem. For example, if you see WebSphere MQ error messages or reason codes in WebSphere Application Server messages and logs, refer to the WebSphere MQ Messages document at <http://publibfi.boulder.ibm.com/epubs/pdf/amqzao05.pdf>.
8. If the information obtained in the preceding stages is still inconclusive you should explore further, for example by enabling the application server debug trace to provide a detailed exception dump.

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 transaction management panels
2. Using methods on the messaging engine MBean

You must first attempt to resolve the indoubt transaction by using the application server transaction management MBean interfaces. These are documented in Managing active and prepared transactions by using wsadmin scripting. Use the scripts for all application servers that might 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 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 MBean:

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

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 enter -1 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.

Procedure

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 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 MBean to check.

Restoring a data store and recovering its messaging engine

When a failure occurs that 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 Service integration 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 undertake further restarts as usual.

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 about the JVM System output file and how to view it, see Viewing JVM logs.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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

Procedure

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 by using the backupConfig command (see Backing up and restoring administrative configuration files). 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.

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

- b. Invoke the start command, with 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 that 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.

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 tried again to connect to the log file but has reached its retry limit and stopped.	<ul style="list-style-type: none">• Check that no other instance of this or any other messaging engine is running that uses 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 tried again to connect to the log file but has reached its retry limit and stopped.	<ul style="list-style-type: none">• 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 across 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.	<ul style="list-style-type: none">• 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 across 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 might 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 required 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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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

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 162
- “Avoiding failover problems when you use DB2 v8.2 with HADR as your data store” on page 162

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.	<ul style="list-style-type: none">• Check that connectivity to the database is possible when 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. <p>If you are using DB2 for z/OS, you can examine the DB2 Idle Thread Timeout.</p>

Symptom	Cause	Solution
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.	<ul style="list-style-type: none"> 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. If you are using DB2 for z/OS, you can examine the DB2 Idle Thread Timeout. You might have 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.	<ul style="list-style-type: none"> 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.
The messaging engine starts but then stops and writes error message CWSIS1519.	The messaging engine has lost its lock on the data store.	<ul style="list-style-type: none"> 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.

Fixing storage allocation error caused by DB2 Universal JDBC Driver on z/OS

When using DB2 Universal JDBC type 2 driver on z/OS, the messaging engine fails to start. This is a known issue and can be solved by tuning the JDBC driver.

About this task

In the address space of Control Region Adjunct (CRA), you might see text "Storage Allocation Error", for example, in

BB000220E: [SB6NLA1:SB6NLA1.server1-SB6NLA1] CWSIP0002E: An internal messaging error occurred in com.ibm.ws.sib.processor.impl.MessageProcessor, 1:1469:1.365, com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E: The data source has produced an unexpected exception: com.ibm.db2.jcc.t2zos.y: [IBM/DB2][T2zos/2.5.48]T2zosPreparedStatement.readPrepareDescribeOutput_:_processDescribeOutput:1563:**Storage Allocation Error** at com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink.readDataFromPersistence (AbstractItemLink.java:2487) at com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink._restoreItem(AbstractItemLink.java:639)

Procedure

1. You can avoid this error by performing one of the following steps:
 - a. Set the JDBC driver customer property `fullyMaterializeLobData` to *false*. You can do this in the administrative console: **Resources** > **JDBC Providers** > *Your JDBC provider* -> **Data sources** -> *Your data source* > **Custom properties** > `fullyMaterializeLobData`.
 - b. Use the Universal JDBC provider of *type 4* driver.
2. Restart the application server.

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:

Procedure

- 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 4 KB.
 - 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, see *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-synchronous 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).

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:

Procedure

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 more messages that exist on a message point for a selected bus destination or messaging engine.


About this task

You should not usually have 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:

Procedure

1. List the messages on the message point.

2. In the content pane, select the check box next to each message that you want to delete. Alternatively, you can select all messages in the list by clicking **Select all items**  .
3. Click **Delete**.

Results

The selected messages are removed from the list.

Troubleshooting service integration message problems

There are tasks that can 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 nonpersistent messages are being discarded”
- “Investigating why a queue is full” on page 165
- “Investigating why a topic space is full” on page 166
- “Investigating why point-to-point messages are not arriving” on page 168
- “Investigating why point-to-point messages are not being consumed” on page 173
- “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181

Understanding why best effort nonpersistent messages are being discarded

A reliability level of *best effort nonpersistent* 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 nonpersistent messages are being discarded by a running system, and you want to understand the possible causes. For more information about *best effort nonpersistent* and other message reliability levels, see [Message reliability levels - JMS delivery mode and service integration quality of service](#).

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.

- A non-transactional message-driven bean generates an exception, and therefore does not complete. For more information, see the *Messages discarded in normal operation* section of Message reliability levels - JMS delivery mode and service integration quality of service.

Investigating why a queue is full

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.

About this task

To investigate why a queue on a service integration bus is full, complete the following steps:

Procedure

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 `../ae/AppsFromSIBRefs_DetailForm.dita` 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, complete the following checks for each message point the message might 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 through a remote queue point. On the producer messaging engine, click **Remote queue points** and then click the queue point that represents the consumer 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, complete 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 166. 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.

Procedure

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.

Procedure

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 might 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

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 might 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.

About this task

To investigate why a topic space on a service integration bus is full, complete the following steps:

Procedure

1. Click **Service integration -> Buses -> bus_name -> [Destination resources] Destinations** to display a list that 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, complete the following checks for each mediation point the message might 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 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 166.
7. If the publishing and subscribing applications are connected to different messaging engines, the messages are being routed through a remote queue point. On the publisher messaging engine, click **Remote publication points** and then click the publication point that represents the subscriber 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, complete 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 166. 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.

Procedure

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.

Procedure

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 might 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

There are a set of checks that you can carry out to investigate why point-to-point messages are not arriving at a destination on a service integration bus.

Before you begin

Complete 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 might 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 nonpersistent messages are being discarded” on page 164 for a list of possible causes.
- Check the priority of the messages. If the priority is low, higher priority work might be delaying the messages.
- Check the system environment, for example, a busy CPU might cause delayed messages.
- Examine the error logs for exceptions.

About this task

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, investigate the problem by completing the following steps:

Procedure

1. Click **Service integration -> Buses -> bus_name -> [Destination resources] Destinations** to display the destinations on the relevant bus. Click on the destination and ensure 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 166.
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 170 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 have to find where the message points for a destination are located.

About this task

You might have to undertake this task as part of problem determination, to find out on which messaging engine a message point is located. If the destination is a queue with multiple queue points, or it is mediated by using multiple mediation points, complete the problem determination for each message point that the message might have been sent to or consumed from.

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.

Procedure

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 destination is a mediated queue, there is 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, there is a publication point localized to every messaging engine in the bus. When a topic space is not mediated, service integration delivers messages directly to the publication point situated on the same messaging engine that the producing application is connected to. When the topic space is mediated, in the same way as a queue, service integration directs messages to the mediation point or points and then to the publication point that is co-located with that mediation point.

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

There are a set of checks that you can carry out 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 168, which contains preliminary checks and investigative tasks that you should carry out before proceeding with this task.

About this task

You should undertake this task as part of “Investigating why point-to-point messages are not arriving” on page 168. 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. In the following diagram, a bus contains three messaging engines, ME1, ME2 and ME3. The producing application is connected to ME1 and the consuming application is connected to ME3. The messages are produced to ME1 and are routed from ME1 to ME3 through ME2. This scenario is only concerned with ME1 and ME2. ME1 hosts a remote message point that represents the message point hosted by ME2. 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.

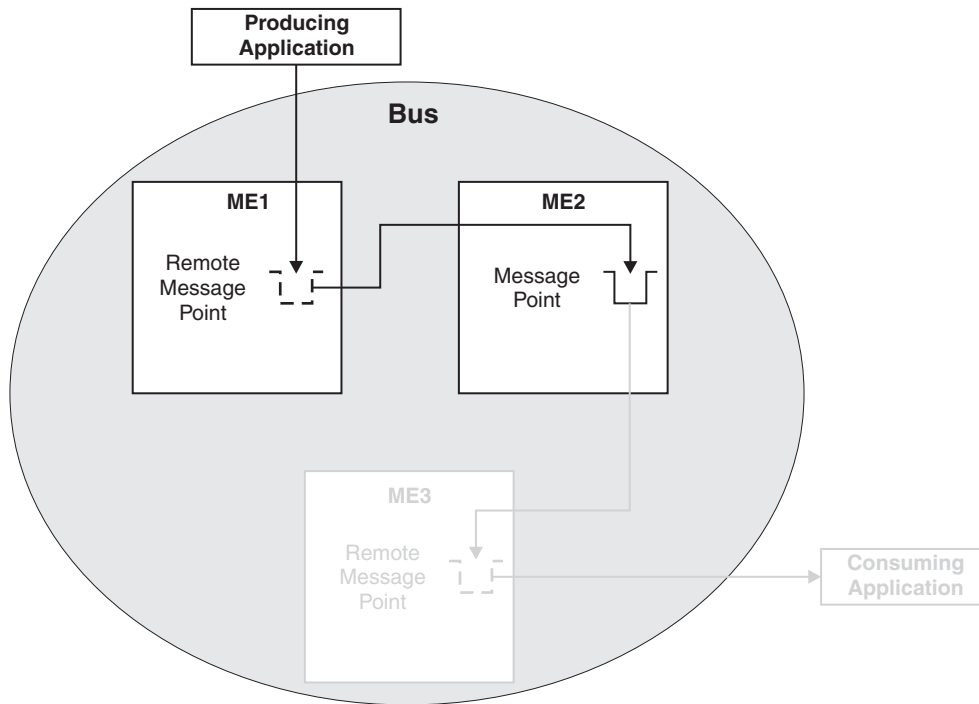


Figure 1. Point-to-point message production by using a remote message point

Procedure

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 166.
 - 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. Complete 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 166.
 - 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 the ME2 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 connected to this messaging engine (see “Determining which messaging engine an application is connected to” on page 166).
 - 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 complete the preliminary checks again.
 5. If the number of current and completed messages are both zero, check that the producing application 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.

Procedure

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.

Procedure

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 might 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

There are a set of checks that you can carry out to investigate why point-to-point messages are not being consumed from a destination on a service integration bus.

Before you begin

Complete the following preliminary checks before starting the investigation:

- Complete 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 might 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 ensure 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 nonpersistent messages are being discarded” on page 164 for a list of possible causes.
- Examine the error logs.

About this task

Complete the following checks 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.

Procedure

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 169.
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 168 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 166.
 - 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” on page 175. 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 178.

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.

Procedure

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

There are a set of checks that you can carry out 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 173 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181, 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 undertake this task as part of either “Investigating why point-to-point messages are not being consumed” on page 173 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181. 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, a bus contains three messaging engines, ME1, ME2 and ME3. The producing application is connected to ME1 and the consuming application is connected to ME3. The messages are routed from ME1 to ME3 through ME2, and are consumed from ME3. This scenario is only concerned with ME2 and ME3. ME3 hosts a remote message point that represents the message point hosted by 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, through a subscription on ME2. a remote subscription point on ME 3. These messaging engines are referred to in the following steps.

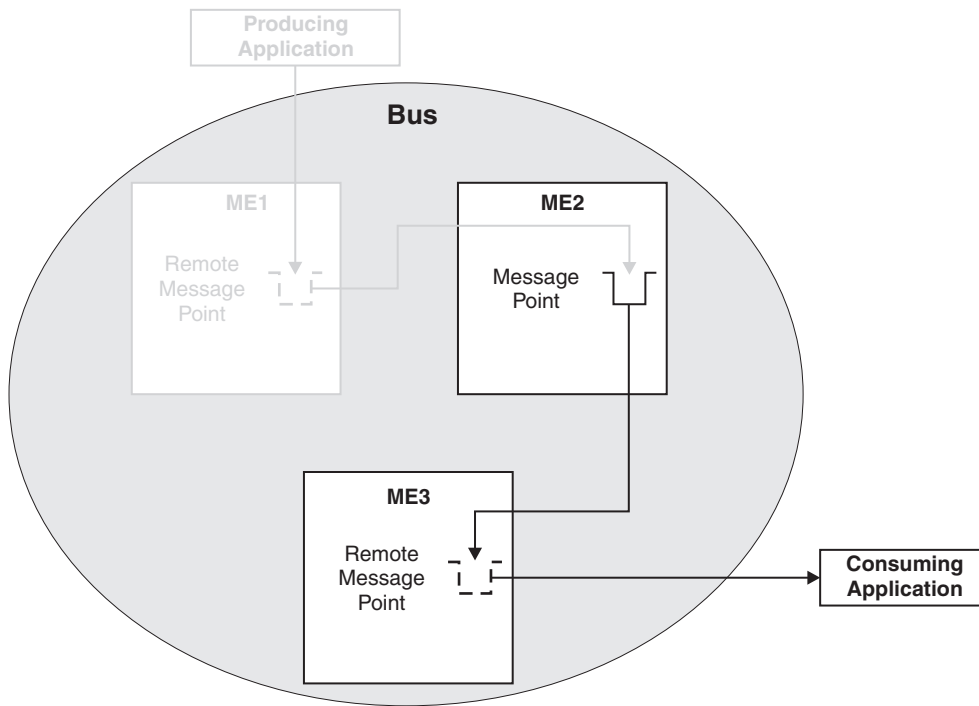


Figure 2. Point-to-point message consumption by using a remote message point

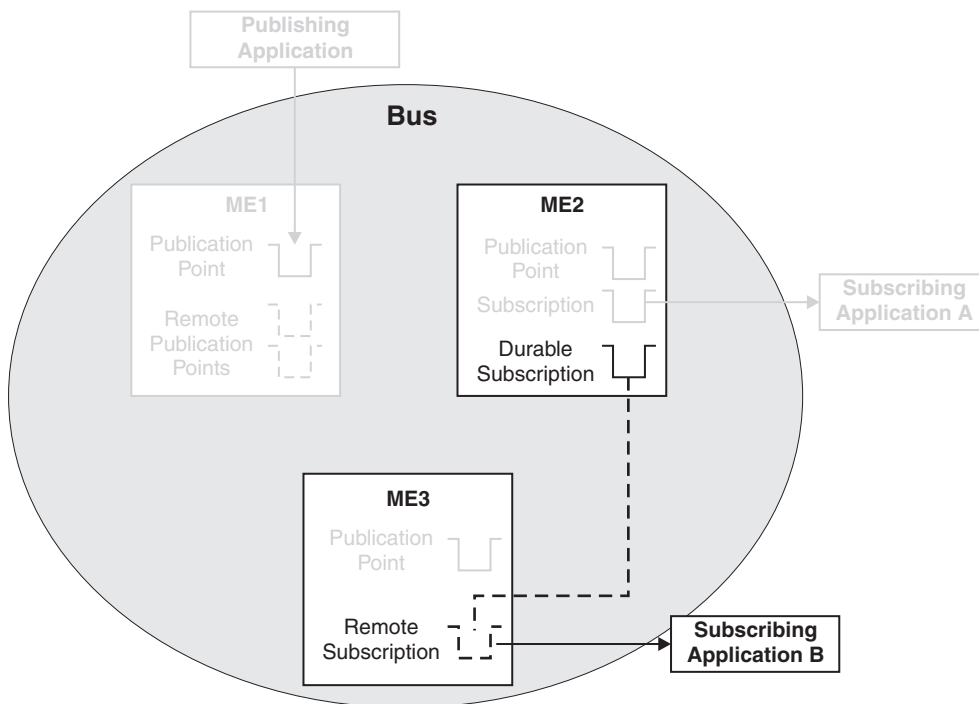


Figure 3. Publish/subscribe messaging by using a remote message point

Procedure

1. If you have followed the steps in “Investigating why point-to-point messages are not being consumed” on page 173 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181 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 might 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 that 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 166.
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 166.

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.

Procedure

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 might 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

There are a set of checks that you can carry out 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 173 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181, 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

Complete this task as part of either “Investigating why point-to-point messages are not being consumed” on page 173 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181. 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, a bus contains three messaging engines, ME1, ME2 and ME3. The producing application is connected to ME1 and the consuming application is connected to ME3. The messages are routed from ME1 to ME3 through ME2, and are consumed from ME3. This scenario is only concerned with ME2 and ME3. ME3 hosts a remote message point that represents the message point hosted by 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, through a subscription on ME2. a remote subscription point on ME 3. These messaging engines are referred to in the following steps.

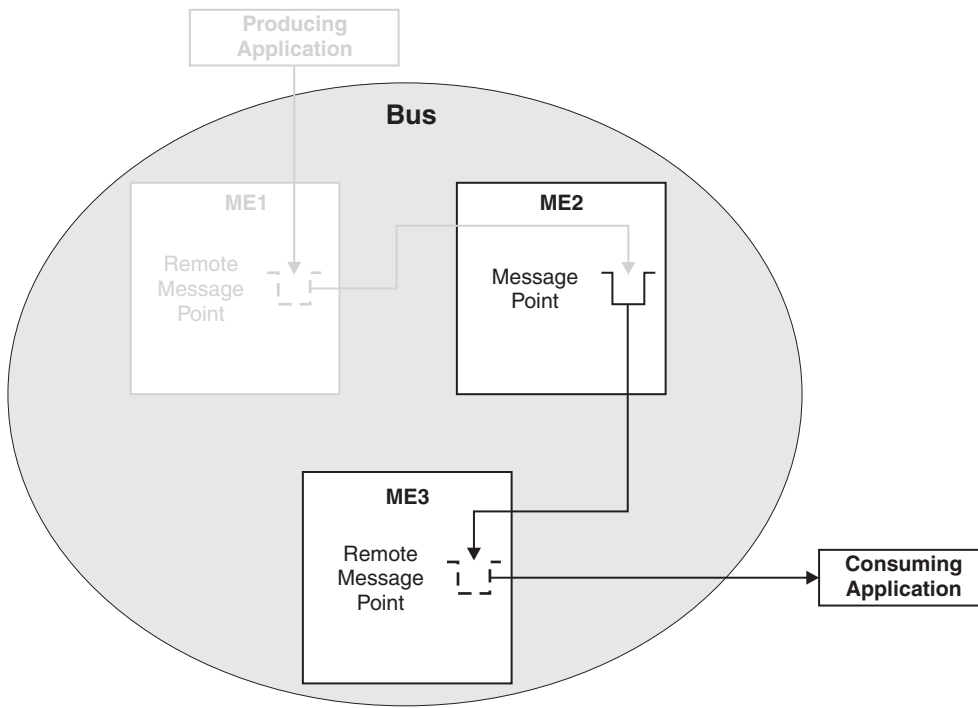


Figure 4. Point-to-point message consumption by using a remote message point

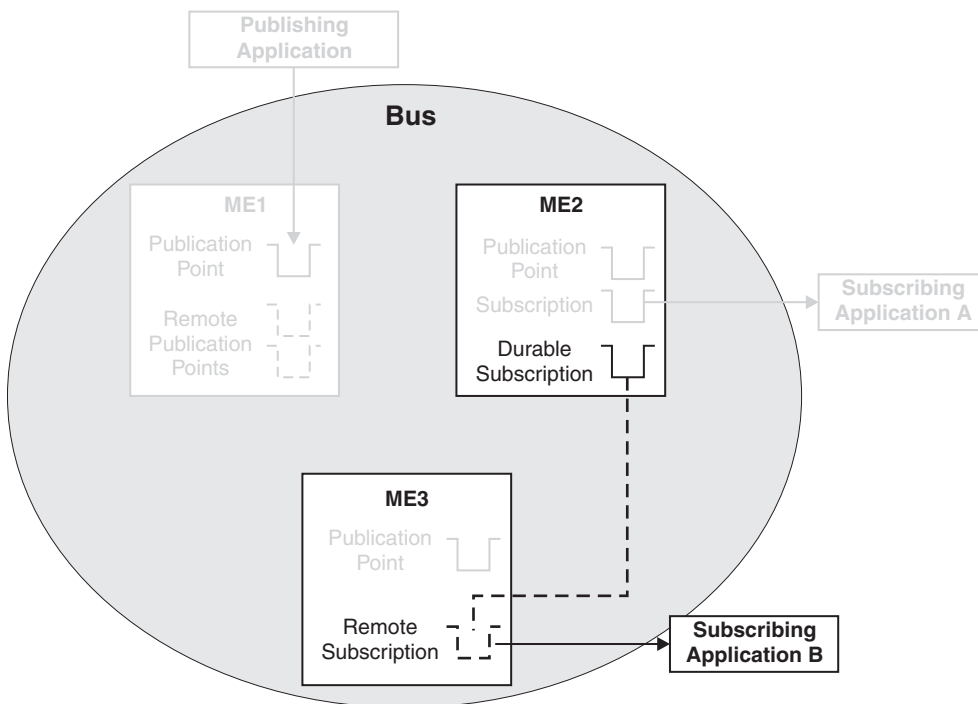


Figure 5. Publish/subscribe messaging by using a remote message point

Procedure

1. If you have followed the steps in “Investigating why point-to-point messages are not being consumed” on page 173 or “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181 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 connected to ME2.
 - Check that the application did not produce any errors that might 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 might 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 166.
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 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 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.

Procedure

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 might 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

There are a set of checks that you can carry out to investigate why publish/subscribe messages are not arriving at a subscription on a service integration bus.

Before you begin

Complete 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 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 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 might 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 nonpersistent messages are being discarded” on page 164 for a list of possible causes.
- Examine the error logs for exceptions.

About this task

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

Procedure

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 166.
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.

Procedure

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

There are a set of checks that you can carry out 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 181, which contains preliminary checks and investigative tasks that you should carry out before proceeding with this task.

About this task

Complete this task as part of “Investigating why publish/subscribe messages are not arriving at a subscription” on page 181. 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, a bus contains three messaging engines, ME1, ME2 and ME3. The publishing application is connected to ME1 and subscribing applications are connected to ME2 and ME3. ME1 hosts remote publication points that represent the publication points hosted by ME2 and ME3. In Figure 2, a bus contains three messaging engines, ME1, ME2 and ME3. The publishing application is connected to ME1 and the subscribing applications are connected to ME2 and ME3. ME1 hosts remote publication points that represent the publication points hosted by ME2 and ME3. Subscribing application B is connected to ME3 and receives publications from ME1 through a remote subscription on ME2. The messaging engines are referred to in the following steps.

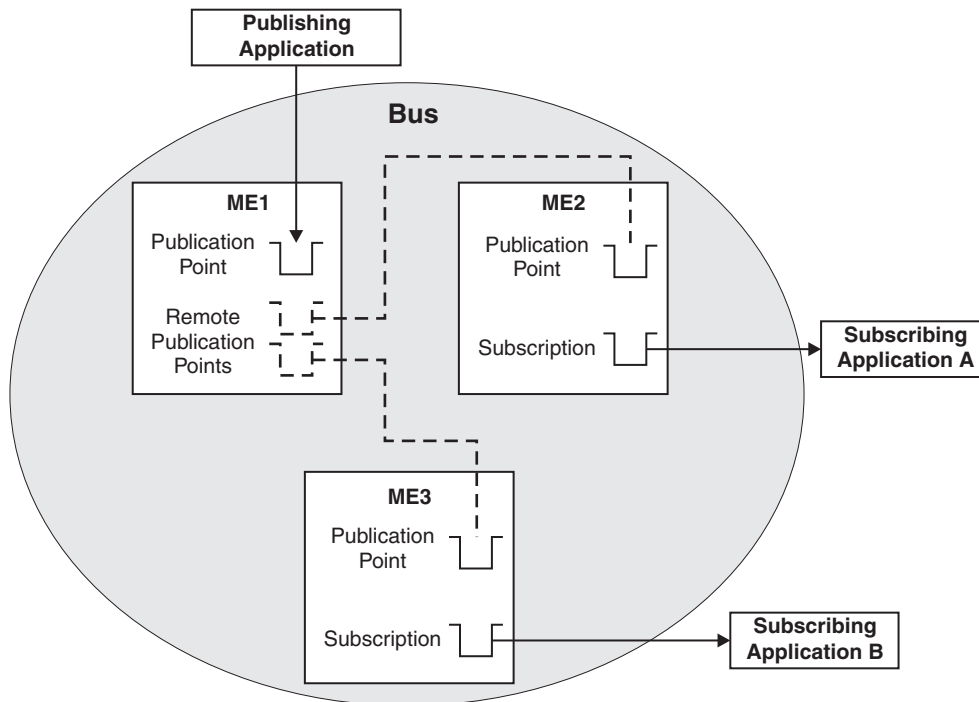


Figure 6. Point-to-point message production by using a remote message point

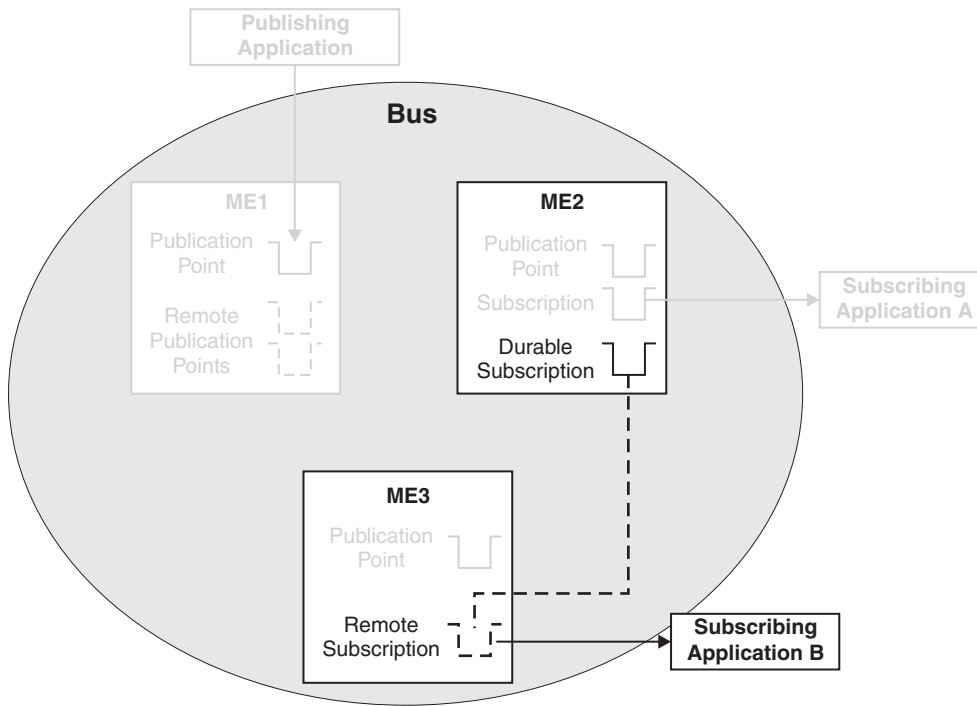


Figure 7. Publish/subscribe messaging by using a remote message point

The following steps apply to both the above scenarios.

Procedure

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 topic is listed. If the topic is not listed, complete 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 166.
 - 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 166.
 - 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. Complete 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 166.
 - 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 the ME2 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 connected to this messaging engine (see “Determining which messaging engine an application is connected to” on page 166).
 - 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 complete the preliminary checks again.
7. If the number of current and completed messages are both zero, check that the producing application 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 that has a remote subscription, you must also investigate the flow of messages between ME2 and ME3, by 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, complete 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 166.
 - 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 175. 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 178.

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.

Procedure

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 might 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.

Procedure

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.

Chapter 15. Mediation thread pool properties

The table below describes the default thread pool properties for the `mediationsThreadPool` object for a messaging engine.

Property name	Description	Data type	Comment
<code>minimumSize</code>	Specifies the minimum number of threads to allow in the pool.	integer	Default value is 1.
<code>maximumSize</code>	Specifies the maximum number of threads to allow in the pool.	integer	Default value is 5.
<code>isGrowable</code>	Can the pool grow beyond the <code>maximumSize</code>	boolean	true or false. Default is false.
<code>inactivityTimeout</code>	Specifies the number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.	integer, units milliseconds.	Default value: 3500

Chapter 16. Interoperating with WebSphere MQ: Troubleshooting tips

Use this set of specific tips to help you troubleshoot problems when using the WebSphere MQ link or WebSphere MQ server components of the default messaging provider to interoperate with WebSphere MQ.

Tips for WebSphere MQ link:

- “The WebSphere MQ link channels do not start.”
- “Messages sent across a WebSphere MQ link are not delivered” on page 192.
- “An application server cannot shut down if a WebSphere MQ link sender channel is waiting for its disconnect interval to expire” on page 192.

Tips for WebSphere MQ server:

- “When a JMS application sends a message to a WebSphere MQ server, a long list of internal error exception messages is issued, including the CWSJP0019E message” on page 193.

The WebSphere MQ link channels do not start

Note: Error messages appear in the SystemOut.log file or, if you have turned on tracing, in the trace.log file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. Read the "Using HPEL to troubleshoot applications" topic in the Troubleshooting section for more information on using HPEL.

1. Verify that the channel names specified on the WebSphere MQ link sender channel and/or the MQLinkReceiver definitions match those specified on the sender and/or receiver channel definitions in the WebSphere MQ network.

Channel names are case sensitive.

2. Verify that the channel sequence numbers are not out of step. If they are, then the channel will remain in a state of retry until the sequence numbers have been reset.

For a WebSphere MQ link sender channel, you can reset the sequence number to 1 by using the WebSphere MQ link sender channel administrative pages. This passes a reset instruction to the WebSphere MQ receiver channel. You can optionally reset the WebSphere MQ receiver channel to a value that matches the WebSphere MQ link sender channel. This does not result in any data being passed to the WebSphere MQ link sender channel, and can be used to resolve sequencing issues.

For a WebSphere MQ link receiver channel, you have to reset the sequence number in WebSphere MQ through the WebSphere MQ sender channel. If you are using the WebSphere MQ Explorer on a Windows system, you can right click on the channel and select **All Tasks > Reset**.

Look for messages CWSIC3011E, CWSIC3015E.

3. Verify that both ends of the channel have been defined and configured correctly. It is possible that the channel at the remote end is currently in a stopped state and therefore is currently unavailable. Start the channel at the remote end if possible.

Look for messages CWSIC3018E, CWSIC3113E, CWSIC3114E, CWSIC3236E.

4. Verify that the channel sequence number wrap values are the same at both ends of the channel.

Look for message CWSIC3010E.

5. Verify that the WebSphere MQ link sender channel is not in an indoubt state. Resolve the channel if required. The channel is resolved by WebSphere MQ. On Windows, if you are using the WebSphere MQ Explorer, you can right click on the channel and select All Tasks>Resolve.
Look for message CWSIC3065E.
6. Verify that the listeners have been started, and are listening on the correct ports. By default, service integration listens on port 5558 for inbound connections, and the WebSphere MQ network listens on port 1414.

Messages sent across a WebSphere MQ link are not delivered

Note: Error messages appear in the SystemOut.log file or, if you have turned on tracing, in the trace.log file. You can also look for equivalent messages in the WebSphere MQ error logs (or trace files if you have turned on tracing in the WebSphere MQ network).

1. If you are sending messages from a service integration bus to a WebSphere MQ network, it is possible that the messages are stored on the service integration bus and waiting to be delivered, but that the WebSphere MQ link sender channel has not been started or is in a retry state.

Verify that the WebSphere MQ link sender channel is started and in running state.

2. If you are sending messages from a WebSphere MQ network to a service integration bus, it is possible that the messages are stored on the transmission queue in the WebSphere MQ network and waiting to be delivered, but that the sender channel in the WebSphere MQ network has not been started or is in a retry state.

Verify that the sender channel in the WebSphere MQ network is started and in running state.

3. It is possible that the messages could not be processed or delivered to the target destination and hence they have been placed either on an exception destination on the service integration bus, or on the dead letter queue in the WebSphere MQ network. Verify that the WebSphere MQ Link on the messaging engine is configured properly with the correct foreign bus, queue manager name (service integration bus), sender channel and receiver channel. The sender channel on the WebSphere MQ Link should match the receiver channel on WebSphere MQ. The receiver channel on the WebSphere MQ Link should match the sender channel on WebSphere MQ.

Look for messages CWSIC3096I, CWSIC3098I, CWSIC3200E, CWSIC3209E.

Check the exception destinations and the dead letter queue. It is possible that the target destination has not been defined, or is full in which case, determine why messages are not being processed from the target destination.

4. It is possible that the target destination and the exception destination and/or the dead letter queue are full and that subsequent persistent messages cannot be safely delivered. Under these circumstances the channel is stopped to avoid any loss of messages.

Look for message CWSIP0291W.

Determine why messages are not being processed from the target destination.

5. It is possible that the target destination and the exception destination and/or dead letter queue are full and that subsequent nonpersistent messages are discarded.

Check the persistence of messages being generated by your applications.

6. It is possible that the channel has stopped because the remote system cannot accept messages for some reason.

Look for message CWSIC3080E.

An application server cannot shut down if a WebSphere MQ link sender channel is waiting for its disconnect interval to expire

If a WebSphere MQ link sender channel does not have any messages to deliver, it waits for its specified disconnect interval before timing out. If the application server is shut down while a WebSphere MQ link sender channel is in a wait state, the application server waits for the WebSphere MQ link sender channel to time out before shutting down. A long disconnect interval might delay the server shutdown.

If the application server shutdown is delayed by a WebSphere MQ link sender channel in a wait state, you have two options:

- Attempt to put a message onto the transmission item stream for the WebSphere MQ link sender channel. Note that this might not take the channel out of its wait state if the application server shutdown is already in progress
- Force the termination of the application server process.

To reduce possible delays during application server shutdown, you can specify a smaller value for the disconnect interval. Note that a disconnect interval of 0 indicates an indefinite wait. For more information about setting the disconnect interval for a WebSphere MQ link sender channel, see [Adding or modifying a WebSphere MQ link sender channel](#).

When a JMS application sends a message to a WebSphere MQ server, a long list of internal error exception messages is issued, including the CWSJP0019E message

This occurs when a WebSphere MQ server is configured to connect to an unsupported version of WebSphere MQ.

In this situation, any attempt by a JMS application to send a message to a service integration bus destination that is defined to a WebSphere MQ server bus member results in a long list of exception messages. The CWSJP0019E message indicates that it is a version problem:

```
com.ibm.ws.sib.remote.mq.exceptions.CorruptRMQSessionException:  
CWSJP0019E: An attempt to connect to WebSphere MQ using the information that is  
provided by the WebSphere MQ Server bus member MQServer1-BUS1 resulted in a  
connection to a WebSphere MQ queue manager running on version MQCMDL_LEVEL_600  
on platform MQPL_WINDOWS_NT. This configuration is not supported. Destinations  
that are assigned to the WebSphere MQ Server bus member are not accessible.
```

Verify that you have configured the WebSphere MQ server to interoperate with a supported version of WebSphere MQ. For interoperation with WebSphere Application Server Version 7.0 or later, the version of WebSphere MQ must be WebSphere MQ for z/OS Version 6 or later, or WebSphere MQ (distributed platforms) Version 7 or later.

Chapter 17. Bus members troubleshooting tips

Use this set of specific tips to help you troubleshoot problems with bus members.

- Error 500 when adding a new member to a service integration bus.

Error 500 when adding a new member to a service integration bus.

When using the WebSphere Application Server administrative console to add a server or server cluster as a new member of a service integration bus, you can receive an Error 500 message, with the additional message “An error occurred while processing request: /ibm/console/sIBusMemberCollection.do”. In addition, the SystemOut log contains a null pointer exception for the stepsLayout.jsp; for example:

```
[9/21/04 14:35:18:282 CDT] 00000031 ServletWrapper E SRVE0068E: Could not invoke
the service() method on servlet /secure/layouts/stepsLayout.jsp. Exception thrown :
java.lang.NullPointerException
```

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

This occurs in situations where your firewall product is configured to remove the private header information (Referer) from requests, because the WebSphere Application Server administrative console requires the header to contain the referer.

To solve this problem, you can either configure your firewall product to allow the private header information (Referer) in requests or disable the firewall product. For example, for information about configuring ZoneLabs to allow the referer in requests, see the *Private Header Information (Referer)* article at http://www.hotcomm.com/FAQ/FAQ_ZA.asp#referer.

Chapter 18. Messaging engine troubleshooting tips

Use this set of specific tips to help you troubleshoot problems with service integration messaging engines.

- “Messaging engine start fails because runtime is not yet initialized”
- “Messaging engine does not start up with a DB2 Universal JDBC type 2 driver”
- “Messaging engine cannot start up because of a known error in the Informix JDBC Driver 3.00JC1” on page 198
- “Problem determination for a data store” on page 198
- “Messaging engines cause database contention messages” on page 199
- “Possible causes of the XAResourceNotAvailableException exception and how to take appropriate action” on page 199
- “Problems when you re-create a service integration bus” on page 200
- “Problems communicating with foreign buses” on page 200
- “Problems when attempting to communicate with a renamed foreign bus” on page 200
- “Possible causes of a JMSEException with a wrapped SILimitExceeded exception” on page 201
- “Corruption problems on system restarts” on page 201
- “Retrieving the status of messaging engines in the administrative console” on page 202
- “Enabling an application to be started before a required messaging engine has started” on page 202
- “Channel framework messages appear during server startup” on page 202
- “Messaging engine failover is not supported for mixed version clusters containing Version 6 servers” on page 203

Messaging engine start fails because runtime is not yet initialized

A messaging engine fails to start and the following error is displayed in the WebSphere Application Server administrative console:

```
The messaging engine <name> cannot be started as there is no runtime initialized for it yet, retry the operation once it has been initialized. If dynamic configuration reload is enabled for this bus, then the servers must be restarted.
```

Before trying to start the messaging engine again, make sure that you have restarted the server. For the runtime to initialize successfully, the application server must be started.

To find out whether a start up problem is preventing the messaging engine runtime from initializing, check for error messages in the SystemOut.log of the hosting server.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Messaging engine does not start up with a DB2 Universal JDBC type 2 driver

When attempting to use the DB2 Universal JDBC type 2 driver to store data on the z/OS platform, the messaging engine does not start up and “Storage Allocation Error” messages similar to the following message might appear in the WebSphere Application Server SystemOut.log file:

```
BB000220E: [SB6NLA1:SB6NLA1.server1-SB6NLA1] CWSIP0002E: An internal messaging error occurred in com.ibm.ws.sib.processor.impl.MessageProcessor, 1:1469:1.365, com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E: The data source has produced an unexpected exception: com.ibm.db2.jcc.t2zos.y: [IBM/DB2] [T2zos/2.5.48]T2zo
```

```
sPreparedStatement.readPrepareDescribeOutput_;processDescribeOutput:1563:Storage Allocation Error at com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink.readDataFromPersistence(AbstractItemLink.java:2487) at com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink._restoreItem(AbstractItemLink.java:639)
```

For the z/OS platform, you should use a DB2 Universal JDBC type 4 driver. If you must use the DB2 Universal JDBC type 2 driver, complete the following steps:

1. Use the administrative console to navigate to **Resources -> JDBC -> Data sources -> data_source_name -> [Additional Properties] Custom properties**
2. Set the JDBC driver custom property fullyMaterializeLobData to **false**.
3. Save your changes to the master configuration.
4. Restart the application server.

Messaging engine cannot start up because of a known error in the Informix JDBC Driver 3.00JC1

When attempting to use the Informix JDBC driver 3.00JC1 to store data, the messaging engine cannot start up and the following error message might appear in the WebSphere Application Server SystemOut.log file:

```
00000022 SibMessage E [RetireBus:retire_web.000- RetireBus] CWSIS0002E:
The messaging engine encountered an exception while starting.
Exception: com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E:
The data source has produced an unexpected exception: java.sql.BatchUpdateException:
Unique constraint (informix.u114_62) violated.
00000022 SibMessage E [RetireBus:retire_web.000- RetireBus] CWSID0035E:
Messaging engine retire_web.000-RetireBus cannot be started;
detected error reported during com.ibm.ws.sib.msgstore.impl.MessageStoreImpl start()
00000022 SibMessage E [RetireBus:retire_web.000- RetireBus] CWSID0027I:
Messaging engine retire_web.000-RetireBus cannot be restarted because a
serious error has been reported.T]
00000022 SibMessage I [RetireBus:retire_web.000- RetireBus] CWSID0016I:
Messaging engine retire_web.000-RetireBus is in state Stopped.
```

There is a known defect (PTS 172471) in the Informix JDBC Driver 3.00JC1. To avoid this error, upgrade the Informix JDBC Driver to 3.00JC2.

Problem determination for a data store

You can create a dump, in reduced form, of the data in the data store for a messaging engine. The output is intended for use by IBM Service personnel. Contact your support organization for information about how to run the command.

If there is a problem with the data in the data store, it can be hard to diagnose from the trace output. However, you can create a dump, in XML format, of the data in the data store. This makes diagnosis easier because it is a human readable representation that can be transformed to other formats as required. You can create a data store dump by typing the following command in the wsadmin tool:

- Using Jython:

```
AdminControl.invoke(AdminControl.queryNames("type=SIBMessagingEngine,
name=messagingengine,*"),
"dump", "com.ibm.ws.sib.msgstore.*")
```

- Using Jacl:

```
AdminControl invoke [AdminControl queryNames type=SIBMessagingEngine,
name=messagingengine,*]
dump com.ibm.ws.sib.msgstore.*
```


The dump is created as an XML file in the \$WAS_HOME/logs/server1 directory. The file is named according to the format: *messaging_engine_nameUUIDtimestamp.xml*

The format of the file is illustrated in the following example:

```
<MessageStore>
  <itemStreams>
    <ItemStreamLink id="0" state="Available">
      <class>com.ibm.ws.sib.msgstore.ItemStream</class>
      <priority>5</priority>
      <canExpireSilently></canExpireSilently>
      <storageStrategy>STORE_NEVER</storageStrategy>
      <expiryTime>0</expiryTime>
      <sequence>0</sequence>
      <tranID>null</tranID>
      <tickValue>0</tickValue>
    </ItemStreamLink>
    <ItemStreamLink id="2" state="Available" refCount="3" refCountDecreasing="false">
      <class>com.ibm.ws.sib.msgstore.Item</class>
      <priority>5</priority>
      <canExpireSilently></canExpireSilently>
      <storageStrategy>STORE_NEVER</storageStrategy>
      <expiryTime>0</expiryTime>
      <sequence>1</sequence>
      <tranID>null</tranID>
      <tickValue>0</tickValue>
    </ItemStreamLink>
  </itemStreams>
</MessageStore>
```

Messaging engines cause database contention messages

When a messaging engine uses a data store for the message store, if the same messaging engine is accidentally started twice, a database contention message is displayed:

```
CWSIS1546I: The messaging engine, ME_UUID={0}, INC_UUID={1}, has lost
an existing lock or failed to gain an initial lock on the data store.
```

To resolve the problem:

- Check for problems with the database, for example, the database is unavailable.
- Check for problems with the network. For example, if the network is overloaded, two application servers might be able to connect to the database, but might not be able to connect to each other, which might cause resource coordination problems.
- If you have a service integration configuration that provides high availability or workload sharing, check that the appropriate resources are configured correctly. For example, check the messaging engines, the core group policies for those messaging engines, and the match criteria that associate each core group policy with a messaging engine. See *Configuring high availability and workload sharing of service integration*.

Possible causes of the XAResourceNotAvailableException exception and how to take appropriate action

When the deleteNode command is used for a node that hosts messaging engines, those messaging engines are deleted. When new messaging engines are re-created following the addNode command, they have different identifiers and so during transaction recovery it is not possible to connect to the old messaging engines. A message identifying the XAResourceNotAvailableException exception is generated in the SystemOut.log file for each server that hosts a messaging engine.

To solve this problem, you must follow the procedure described in “Resolving indoubt transactions” on page 155.

The XAResourceNotAvailableException exception can also be thrown when a server in a cluster bus member fails over. In this case, no operator intervention is required to recover and resolve transactions.

Problems when you re-create a service integration bus

If you delete a service integration bus, and later create a new bus with the same name, the messaging engine fails to start and messages such as the following are generated in SystemOut.log:

```
[8/11/04 21:55:01:439 CDT] 0000000f SibMessage I
[LateBus:xyzsun15.server1-LateBus] isAlive: MessagingEngine suffered common mode error.
Correct error (see logs) and restart server.
[8/11/04 21:55:01:468 CDT] 0000000f SibMessage I
[LateBus:xyzsun15.server1-LateBus] isAlive: MessagingEngine will be stopped
because of common mode error.
No failover will occur.
[8/11/04 21:55:01:493 CDT] 0000000f SibMessage I
[LateBus:xyzsun15.server1-LateBus] Messaging Engine
xyzsun15.server1-LateBus not in state from which stop is valid: Starting
[8/11/04 21:55:01:513 CDT] 0000000f SibMessage I
[LateBus:xyzsun15.server1-LateBus] isAlive: MessagingEngine stopped because
of common mode error. Correct error (see logs) and restart server.
[8/11/04 21:57:01:431 CDT] 0000000e SibMessage I
[LateBus:xyzsun15.server1-LateBus] isAlive: MessagingEngine suffered
common mode error.
Correct error (see logs) and restart server.
```

The messaging engine failed to start because the database directory for the messaging engine still exists after deletion of the bus and you must manually remove it. To delete the Apache Derby database for a non-existent messaging engine, you must delete the database directory that is located in *profile_root*/databases/com.ibm.ws.sib, where *profile_root* is the directory in which profile-specific information is stored.

You must stop WebSphere Application Server before you can delete the database files.

For other databases, you can either delete all of the rows from the data store tables or you can drop all of the data store tables. These tables are in the schema that you configured for the data store. For a list of the tables, see Data store tables.

For more information, see Data store life cycle.

Problems communicating with foreign buses

To enable communication between buses, a foreign bus and a service bus integration link must be created. On the first bus, the name of the foreign bus must match the name of the second bus that becomes a foreign bus, and the name of the foreign bus for this second bus must match the name of the first bus. The service integration bus link must have the same name on both buses.

You may see the following type of error if your configuration is not correct, for example because the service integration bus links do not match:

```
SibMessage E [TechBus:TechCluster.000-TechBus]
CWSIT0057E: The inter-bus connection BookstoreBus failed in the remote
messaging engine on host aixp401.rchland.ibm.com with reason:
CWSIT0067E: Inter-bus connection BookstoreBus in bus BookstoreBus
is not available.
```

Problems when attempting to communicate with a renamed foreign bus

The administrative console panel used for configuring the properties of a service integration bus link can also be used to change the foreign bus name that the link is pointing to. However, you must not alter the name of the foreign bus after it has been configured. If you do, any messaging engines that already hold state information about the link will not be able to use the link until the foreign bus name is reset to its previous value.

Possible causes of a JMSException with a wrapped SILimitExceeded exception

When the number of messages held by a destination reaches its limiting threshold, any attempt to send a message to that destination fails with a JMSException with a wrapped SILimitExceeded exception. The destination continues to fail with this exception until the number of messages held by the destination reduces below the limiting threshold.

To obtain an accurate count of the number of available messages, you can monitor the Available Message Count PMI statistic for queue and topicspace destinations. If the number of available messages increases, take action to balance the system. Consider stopping producers from sending new messages until the destination consumes the available messages.

Examine the following list for possible causes and solutions for this problem:

- The high threshold for the destination is too low for the projected number of messages. The destination does not process some messages. The default value for the high threshold is 50000.

Solution: Increase the high threshold for the destination.

- Applications are producing more messages than the destination can process.

The ideal balance is for the number of messages produced and the number of messages consumed to be equal over a period of time. If your system is unbalanced and the producing application sends more messages than the destination can consume, the producing application eventually throws a JMSException.

Solution: Aim for a balance between the number of messages produced and the number of messages consumed.

Tip: The default setting for the Object Request Broker (ORB) thread pool is 100 threads. For some applications, this might allow 100 applications to send messages to the same destination. Consider tuning the ORB thread pool to have a maximum of 10 threads. This lower setting reduces the number of producers that can send messages, which might increase the overall message throughput.

- Applications are processing messages from the destination too slowly.

Solution: It might be necessary to increase the number of messages consumed by the client applications. A destination processes more message when multiple consumers read from that destination.

Consider cloning your application across multiple servers in a non-clustered environment. By default, applications are cloned in a clustered server environment. To enable subscribers in a non-clustered environment, set the **cloned** flag in the TopicConnectionFactory JNDI setting for DurableSubscriptions.

Restriction: This solution is not suitable for applications that require total message ordering.

- Messages have a quality of service attribute that is *better than* best effort nonpersistent.

Solution: Use messages for which the quality of service attribute is best effort nonpersistent. If there are too many messages in the system, the destination discards best effort nonpersistent messages.

Restriction: This solution is not suitable for applications that must receive all messages.

Corruption problems on system restarts

It is possible, although rare, for a messaging engine, destination or link to be corrupted after a restart of the system. If this corruption occurs you will see a message indicating the problem. If the problem lies with

the messaging engine, the messaging engine will not start. If a destination or link is corrupted, the relevant messaging engine will start, but the destination or link will not be usable on that messaging engine.

If you do not know the cause of the problem, contact your IBM service representative to establish the cause before attempting to resolve the situation.

If you know the cause of the problem, for example, you are aware of an issue with your database, resolve it by completing the following steps:

1. Use the administrative console to ensure that the configuration files are synchronized across your system, by navigating to **System administration -> Nodes** then clicking **Full Resynchronize**. This operation can take several minutes to run.
2. If the problem still exists, complete one of the following tasks:
 - Delete the corrupted object and recreate it. Messages produced or received before the corruption occurred will be lost.
 - Restore your system from a backup, see “Restoring a data store and recovering its messaging engine” on page 156. Messages produced or received since the backup was taken will be lost.

Retrieving the status of messaging engines in the administrative console

To be able to retrieve the status of messaging engines, you must be logged into the administrative console with at least monitor authority. If you do not have this authority, the messaging engine status is displayed as "Unavailable", even if the messaging engine has started.

If you are not logged in with the authority needed to retrieve the status of messaging engines, an error message such as the following is logged in the server systemOut log file:

```
[4/20/05 10:49:57:083 CDT] 0000004b RoleBasedAuth A
SECJ0305I: The role-based authorization check failed for admin-Authz
operation SIBMessagingEngine:stateExtended.
The user UNAUTHENTICATED (unique ID: unauthenticated) was not granted any of
the following required roles: administrator, operator, configurator, monitor.
```

Where the user ID shown in the message is the user ID that you used to log in to the administrative console.

Enabling an application to be started before a required messaging engine has started

If an application depends on a messaging engine being available, then the messaging engine must be started before the application can be run. If you want application server to start an application automatically, you should develop your applications to test that any required messaging engine has been started and, if needed, wait for the messaging engine. If this technique is used in a startup bean, then the startup bean method should perform the test and wait work in a separate thread (using the standard WorkManager methods), so that the application server startup is not delayed.

For an example of code to test and wait for a messaging engine, see Applications with a dependency on messaging engine availability.

Channel framework messages appear during server startup

When you start the server, you might see channel framework information messages displayed in the control region adjunct (CRA) process. These messages do not indicate that any errors have occurred and do not require you to take any action.

- The following message is issued because the application that contains the message-driven bean has started before the messaging engine has started.

CWSIV0759W: During activation of a message-driven bean, no suitable active messaging engines were found in the local server on the bus {0}.

When the messaging engines start, another information message confirms this and message handling can then occur.

- The following message is issued because of the asynchronous way in which the z/OS TCP proxy channel starts.

```
CHF0030E: Error starting chain {0} because of exception {1}
```

When the messaging engines start, another information message confirms this and message handling can then occur.

These messages only appear under certain circumstances, for example these messages might appear if you change ports during migration

- The following message might be displayed several times in the control region adjunct process during server startup, even though the connection succeeds on subsequent retries. This message is issued because of the asynchronous way in which the z/OS TCP proxy channel starts and does not indicate that any error has occurred.

```
Trace: 2009/06/17 08:24:41.434 01 t=9C6B58 c=UNK key=P8 (00000011)
```

```
Description: Log Java Message
```

```
Message: CHF0030E: Error starting chain _InboundTCPProxyBridgeService because of exception
```

```
com.ibm.wsspi.channel.framework.exception.RetryableChannelException:
```

```
An exception was thrown when attempting to start the TCPProxyChannel
```

```
com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl
```

These messages might be accompanied by First Failure Data Capture (FFDC) output similar to the following example:

```
Exception = com.ibm.wsspi.channel.framework.exception.RetryableChannelException
```

```
Source = com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.startChainInternal
```

```
probeid = 2577
```

```
Stack Dump = com.ibm.wsspi.channel.framework.exception.RetryableChannelException: An exception was thrown when attempt
at com.ibm.ws.tcpchannelproxy.jfap.impl.TCPProxyInboundChannel.start(TCPProxyInboundChannel.java:153)
at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.startChannelInChain(ChannelFrameworkImpl.java:1410)
at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.startChainInternal(ChannelFrameworkImpl.java:2863)
at com.ibm.ws.channel.framework.impl.WSChannelFrameworkImpl.startChainInternal(WSChannelFrameworkImpl.java:960)
at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.startChainInternal(ChannelFrameworkImpl.java:2794)
at com.ibm.ws.channel.framework.impl.ChannelFrameworkImpl.startChain(ChannelFrameworkImpl.java:2779)
at com.ibm.ws.runtime.component.ChannelFrameworkServiceImpl.startChain(ChannelFrameworkServiceImpl.java:666)
at com.ibm.ws.sib.jfapchannel.framework.impl.ChannelFrameworkReference$TCPProxyBridgeServiceInboundChainStartu
at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1550)
```

```
Caused by: com.ibm.ws.tcpchannelproxy.jfap.NotYetInitializedException: Server is not yet initialized
```

```
at com.ibm.ws.tcpchannelproxy.jfap.TCPProxyBridgeServicesImpl.startListening(TCPProxyBridgeServicesImpl.java:5
```

```
at com.ibm.ws.tcpchannelproxy.jfap.impl.TCPProxyInboundChannel.start(TCPProxyInboundChannel.java:131)
```

```
... 8 more
```

Eventually the following message should be displayed indicating that the z/OS TCP proxy channel has started up correctly:

```
Trace: 2009/06/17 08:24:51.449 01 t=9C6B58 c=UNK key=P8 (13007002)
```

```
ThreadId: 00000003
```

```
FunctionName: com.ibm.ws.channel.framework.impl.WSChannelFrameworkImpl
```

```
SourceId: com.ibm.ws.channel.framework.impl.WSChannelFrameworkImpl
```

```
Category: AUDIT
```

```
ExtendedMessage: BB000222I: CHF0019I: The Transport Channel Service has started chain _InboundTCPProxyBridgeService.
```

Messaging engine failover is not supported for mixed version clusters containing Version 6 servers

A messaging engine that is hosted on a WebSphere Application Server Version 7.0 or later server cannot fail over to a messaging engine that is hosted on a WebSphere Application Server Version 6 server. If you have a cluster bus member that consists of a mixture of Version 6 and later servers, you must make sure the high availability policy is configured to prevent this type of failover.

To prevent failover of a Version 7.0 or later messaging engine to a Version 6 server, configure the high availability policy for the messaging engine so that the cluster is effectively divided into one set of servers for Version 6 and another set of servers for Version 7.0 or later, and the Version 7.0 or later messaging engine is restricted to the servers at Version 7.0 or later. See [Configuring messaging engine failover for mixed version clusters](#).

Chapter 19. Default messaging provider: Troubleshooting tips

Use this set of specific tips to help you troubleshoot problems for JMS messaging with the default messaging provider.

For general tips about troubleshooting problems with WebSphere Application Server messaging, see “Messaging troubleshooting tips” on page 72. This topic provides additional tips specific to the default messaging provider and its use of service integration technologies.

- “A destination becomes full and can no longer receive messages because the existing messages are not being consumed.”
- “A JMS application can no longer send or receive messages”
- “JMS client applications running inside the Java Platform, Enterprise Edition (Java EE) client container fail when invoking the `ConnectionFactory.createConnection` method”

A destination becomes full and can no longer receive messages because the existing messages are not being consumed.

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.

You can use the following administrative console panel to inspect the configuration of the applications and JMS resources that are using the destination: `../ae/AppsFromSIBRefs_DetailForm.dita`.

This panel can help you find the cause of the problem by giving you a high level view of many relevant resources.

A JMS application can no longer send or 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.

You connect your application to the message beans or enterprise beans through the application's deployment descriptor or through code in the application itself. If you connected your application through the deployment descriptor, you can use the following administrative console panel to view the installed business application as whole and inspect the configuration of the JMS resources that are being used by the application: `../ae/AppToSIBRefs_DetailForm.dita`.

This panel can help you find the cause of the problem by giving you a high level view of many relevant resources.

JMS client applications running inside the Java Platform, Enterprise Edition (Java EE) client container fail when invoking the `ConnectionFactory.createConnection` method

When a JMS client application is running inside the Java EE client container, on a machine that is running no other WebSphere Application Server processes, a call to the `ConnectionFactory.createConnection` method can fail with the following error:

```
CWSIJ0005E: An instance of the channel framework service to use for communication cannot be found.
```

Cause

The ConnectionFactory for the default messaging provider has a dependency on the Channel Framework Service. It locates the Channel Framework Service using a lookup in the JNDI namespace. To connect to a naming service, the ConnectionFactory uses an InitialContext object created by using the default constructor.

When the JMS client is running within an application server environment, the InitialContext object is able to successfully connect to the naming service, the Channel Framework Service is located and the call to createConnection completes successfully.

However, when the JMS client is executing within the Java EE client container, the InitialContext object uses the value of the java.naming.provider.url system property, to determine the location of the naming service to connect to. If no value is specified for this property, it attempts to connect to a naming service located at port 2809 on the local client machine. If there is no server running on the client machine, there is no naming service listening on this port on the local machine. This causes the createConnection method to fail.

Solution

A JMS client application can specify the value of the java.naming.provider.url programmatically, by using code of the form:

```
String key = "java.naming.provider.url";
String value = "iiop://some.remote.machine:9810";
System.setProperty(key, value);
```

This code should be run prior to invoking the createConnection method on the ConnectionFactory object.

Alternatively, if you use the launchClient script from the command line to start the Java EE client container, you can specify either of the following command line parameters:

- launchClient <CLIENT EAR> -CCBootstrapHost=some.remote.machine -CCBootstrapPort=981
- launchClient <CLIENT EAR> -CCproviderURL=iiop://some.remote.machine:9810

This ensures that if no provider URL is specified programmatically, then any InitialContext objects default to using the provider URL specified on the command line.

Chapter 20. Tips for troubleshooting mediations

Use this set of specific tips to help you troubleshoot problems with mediations.

To help identify and resolve problems, use the Troubleshooter reference: Messages. It contains information about error 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.

For information about looking at error logs, see Diagnosing problems with message logs.

Tips for common problems

- “Where is my message?”
- How do I enable trace?
- Why was my message not mediated?
- Why does my mediation not run?
- Which error messages relate to mediations?

Where is my message?

If your message is not where you expect it to be, check the following:

- The message had the reliability level Best effort nonpersistent, and has been discarded as a result of constrained system resources. For more information, see Reliability levels.
- The message has been received by another consumer, and has gone from the system.
- The message expiry time has exceeded, and the message has been discarded.
- The mediation returned false, and the message has been discarded. For more information, see Programming mediations .
- The mediation threw an exception, and the message was sent to the exception destination. For more information, see Error handling in mediations.
- The destination to which the message was forwarded has been deleted, and the message has been sent to the exception destination. For more information, see Exception destinations and Error handling in mediations.
- The forward routing path has been modified to behave other than intended, and the message has not been routed to the intended destination. For more information, see Configuring a destination forward routing path.
- There is an implicit loop in the forward routing path, and the message has not arrived at the intended destination.

How do I enable trace?

For more information, see Working with trace.

Why was my message not mediated?

- Check if the destination is mediated. To do this, list the mediation points for the destination. For more information, see Listing mediation points for a bus destination.
- Look in the system.out log for the following error message:

```
CWSIZ0002E: The mediation named {0} that is attached to destination {1}
is defined to use mediation handler list {2}. However this handler list
does not exist.
```

Check for the following common reasons why the handler list does not exist:

- The application containing the handler list has not been installed.
- The application containing the handler list has not been started.

- The name of the handler list is mis-spelled in the mediation definition.
- Check the status of the mediation handler application in the administrative console. If it has the status Stopped, check for error messages in the `system.out` log.
- Check that the message matches the selector and/or discriminator configured on the mediation. To do this:
 1. View the message on the mediation point. For more information, see [Viewing messages for a bus destination](#).
 2. Check the configuration for the mediated destination. For more information, see [Configuring mediation properties](#).

Why does my mediation not run?

If your mediation does not run, check the following:

- The status of the mediation in the administrative console.
- The stop reason on the mediation point. For more information, see [restarting a mediation that has stopped on error](#).
- Error messages in `system.out` log. For more information, see [Diagnosing problems with message logs](#).

Which error messages relate to mediations?

Mediations error messages have the following prefixes:

CWSJU

Errors about message production or consumption.

CWSIZ

Errors about what has happened to the message when it was mediated.

Chapter 21. Service integration bus security: Troubleshooting tips

Use this set of specific tips to help you troubleshoot problems you experience when working with a secure service integration bus.

To help you identify and resolve service integration bus security-related problems, use the WebSphere Application Server trace and logging facilities as described in [Setting up component trace \(CTRACE\)](#).

If you encounter a problem that you think might be related to service integration bus security, 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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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 service integration bus security component is `CWSII`.

The Troubleshooter reference: Messages 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:

- “After you migrate a Version 5.1 application server to WebSphere Application Server Version 7.0 or later, existing web services clients can no longer use SOAP over JMS to access services hosted on the migrated server.”
- “When you try and make a connection by using a user ID in an authorized group, access is denied when using LDAP” on page 210

After you migrate a Version 5.1 application server to WebSphere Application Server Version 7.0 or later, existing web services clients can no longer use SOAP over JMS to access services hosted on the migrated server.

Before you migrated the Version 5.1 application server, no user ID or password was required on the target MQ Series queue. After the application server is migrated to Version 7.0 or later, and to use the default messaging provider (the service integration bus), client requests fail because basic authentication is now enabled. The problem appears as a log message:

```
SibMessage    W    [:] CWSIT0009W: A client request failed in the application
server with endpoint <endpoint_name> in bus your_bus with reason: CWSIT0016E:
The user ID null failed authentication in bus your_bus.
```

In WebSphere Application Server Version 7.0 or later, when you use a service integration bus and WebSphere Application Server security is enabled for the server or cell, by default the service integration bus queue destination inherits the security characteristics of the server or cell. So if the server or cell has basic authentication enabled, then the client request fails.

To resolve the problem, you have three choices (in order of security, from least secure to most secure):

- Disable security.
- For an equivalent level of security to the configuration on Version 5.1, modify the settings for the service integration bus that hosts the queue destination so that bus security is disabled and therefore the bus does not inherit security characteristics from the server or cell.
- For a greater level of security than the configuration on Version 5.1, configure basic authentication on each client that uses the service.

To disable WebSphere Application Server security, refer to Enabling and disabling security using scripting, or Global security settings.

To disable bus security, use the administrative console to complete the following steps:

1. Navigate to **Service integration -> Buses -> *bus_name***.
2. Clear the **Secure** check box.
3. Save your changes.

To configure basic authentication on each client, use either the administrative console or the wsadmin tool. To complete the task by using the wsadmin tool, see Configuring web service client port information using wsadmin scripting and use the WebServicesClientBindPortInfo wsadmin task option. To complete the task by using the administrative console, complete the following steps:

1. Navigate to **Applications -> Application Types -> WebSphere enterprise applications -> *application_name* -> Web Modules or EJB Modules > *module_name* > Web services: Client security bindings**.
2. Click **HTTP basic authentication** to access the “Configuring HTTP basic authentication with the administrative console” panel.
3. Enter the values in the panel.
4. Save your changes to the master configuration.

When you try and make a connection by using a user ID in an authorized group, access is denied when using LDAP

One of the possible causes is the group name, if you are using an Lightweight Directory Access Protocol (LDAP) registry. When you specify the group authorization permissions, the distinguished name (DN) should be used as the group name. If you specify a common name (CN) for the group name users in that group cannot be authorized.

Steps to change the group name from CN to DN depends on where the problem occurred.

- If you have problem connecting to a service integration bus, see Administering the bus connector role, and remove any groups with CN group names, and replace them with DN group names.
- If you have problem sending a message to a destination, see Administering destination roles, and remove any groups with CN group names, and replace them with DN group names.
- If you have problems sending topics to a topic space, see Administering topic space root roles, and remove any groups with CN group names, and replace them with DN group names.
- For any other problems refer to the appropriate section on Administering authorization permissions on how to modify the group name.

Chapter 22. Troubleshooting Session Initiation Protocol (SIP) applications

This page provides a starting point for finding information about SIP applications, which are Java programs that use at least one Session Initiation Protocol (SIP) servlet written to the JSR 116 specification.

SIP is used to establish, modify, and terminate multimedia IP sessions including IP telephony, presence, and instant messaging.

Troubleshooting SIP applications

Use this page to troubleshoot SIP applications.

About this task

SIP container troubleshooting basics

- The Average CPU usage of the system should go no higher than 60%-70%.
- The container should use no more than 70% of the allocated VM heap size. Be sure that the system has enough physical memory to accommodate the VM heap size. Call loads and session timeouts will have a big affect on heap usage.
- The maximum garbage collection (GC) time of the VM on which the container is running should not exceed 500 ms and the average should be less than 400 ms. Verbose GC can be used to measure this and the PMI viewer can be used to view GC times, heap usage, active sessions, etc., in graphical form.

Initial troubleshooting checklist:

Procedure

- Check the listening ports in the configuration.
- Use `netstat -an` to see listening ports.
- Check to see if virtual hosts are defined
- Check to see if host aliases are defined.
- Is an application installed? Is it started?
- For a proxy configuration: Is a default cluster configured? If proxy and server are on the machine, is there a port conflict?

Results

SIP container symptoms and solutions

If the problem is not resolved, check for specific symptoms.

- **Symptom:** Lots of retransmissions, CPU periodically drops to zero, or a `ThreadPoolQueueFullException` exception is seen in the logs

Solution: This is typically a DNS issue caused by Reverse DNS lookups and can be confirmed using a tool like Ethereal. If you do a network capture and send lots of DNS queries that contain an IP address and get back a host name in the response, this could be the problem. Make sure that `nsd` is running if you are on HP or some other platform that requires name service caching. The Microsoft Windows platform does not require name service caching. Another solution is to add host names to the `/etc/hosts` file.

- **Symptom:** Lots of retransmissions, CPU periodically spikes to 100%.

Solution: This is typically due to garbage collection and can be verified by turning on verbose GC (accessible on the admin console) and looking at the length of the GC cycles. The solution here is to enable Generational Garbage Collection by setting the JVM optional args to `-Xgcpolicy:gencon`.

- **Symptom:** Lots of retransmissions, CPU spikes to 100% for long periods of time and Generational Garbage Collection is enabled.

Solution: This is typically due to SIP session objects either not being invalidated or not timing out over a long period of time. One solution is to set the session timeout value in the `sip.xml` of the application to a smaller value. The most efficient way to handle this is for the application to call `invalidate` on the session when the dialog completes (i.e. after receiving a BYE). The following entry in the `SystemOut.log` file will indicate the session timeout value is for each application installed on the container:

```
SipXMLParser 3 SipXMLParser getAppSessionTTL Setting Expiration time: 7 Minutes, For App: TCK back-to-back user agent"
```

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

- **Symptom:** Lots of “480 Service Not Available” messages received from the container when sending new INVITE messages to the SIP container. You will also likely see the following message show up in the `SystemOut.log` when the server is in this state: “LoadManager E LoadManager warn.server.oveloaded”.

Solution: This is typically due to one of the SIP container configurable metrics being exceeded. This includes the “Maximum Application Sessions” value and the “Maximum messages per averaging period” value. The solution is to adjust these values higher.

- **Symptom:** Lots of resends and calls are not completing accompanied by `OutOfMemory` exceptions in the `SystemErr.log`.

Solution: This usually means that the VM heap size associated with your container is not large enough and should be adjusted upwards. You can adjust this value from the admin console.

- **Symptom:** You receive a “503 Service Unavailable” when sending a SIP request to a SIP proxy.

Solution: This usually means there is no default cluster (or cluster routing rule that matches the message) set up at the proxy. This can also happen when the SIP proxy is configured well but the backend SIP containers are stopped or have crashed.

- **Symptom:** You receive a “404 Not Found” when sending a SIP request to a SIP proxy.

Solution: This usually means there is no virtual host set up for the containers that reside in the default cluster. It could also mean that the servers in the proxy's default cluster do not contain a SIP application or that the message does not match one of the applications installed in the default cluster.

- **Symptom:** An “out of memory” type behavior is occurring.

Solution: This may be due to the maximum heap size being set too low. SIP applications can consume a significant amount of memory because the sessions exist for a long call hold time. The maximum heap size of 512 MB does not provide sufficient memory for the SIP traffic workload. Set the maximum heap size for SIP applications to the minimum recommended value of 768 MB or higher.

- **Symptom:** You receive a “403 Forbidden” when sending a SIP request to a SIP container.

Solution: This usually means there is no appropriate SIP application found to handle the received SIP request (no match rule that matched the message).

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:

Procedure

1. Open the administrative console. 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
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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Chapter 23. Troubleshooting Transactions

This page provides a starting point for finding information about Java Transaction API (JTA) support. Applications running on the server can use transactions to coordinate multiple updates to resources as one unit of work, such that all or none of the updates are made permanent.

The product provides advanced transactional capabilities to help application developers avoid custom coding. It provides support for the many challenges related to integrating existing software assets with a Java EE environment. More introduction...

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:

Procedure

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 error log.
3. Check for more messages in additional places.
Check for more messages in other potential message output repositories. For more information about a problem, check the standard output file configured by your administrator. This will contain more error messages and other detailed information about the problem.
4. Check for messages related to the application server 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 RRS logs for any transaction activity involving RRS-compliant resources. IBM WebSphere Application Server for z/OS is capable of supporting both XA and RRS resource managers, and of coordinating a mix of RRSTransactional resource managers and XA capable resource managers under the same global transaction. If your installation uses XA resource managers, RRS resource managers, or a mixture of both, you can use the administrative console to view transaction logs that contain information about all transactions. You can find additional information about RRS transactions by using the RRS panels.
6. Check the hints and tips for troubleshooting transactions.
See the topic about 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

Use these 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”
- “Clean shutdown message is not in the message log”

For messaging problems specific to WebSphere Application Server nodes, see other topics in the information center, such as the topic about messaging troubleshooting tips, and the WebSphere Application Server Support web page.

Peer recovery fails to acquire a lock

If peer recovery of a transaction fails to acquire a file lock that is needed to undertake 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 needed 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=fwsitkaCell101\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 0 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 device 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 have 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 might or might 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 shut down 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 behavior is normal, and is 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 message does not indicate a problem, but it does mean that recovery activity is required when the server restarts.

Before you uninstall the product, you should have a clean shutdown of all application servers so that you avoid data integrity problems.

Ensure that recovery from an RRS or XA resource perspective is not needed

On the z/OS operating system, the clean shutdown message CWWTR0105I is never logged. To ensure that recovery from an RRS or XA resource perspective is not needed, you can restart the application server in recovery mode, in the system in which it is configured. In recovery mode, if there are any outstanding units of recovery (URs), the application server completes the URs, then shuts down. If there are no outstanding URs, the application server starts, then shuts down normally. Therefore, to ensure that all recovery has occurred, restart the server in recovery mode and wait until a normal shutdown.

Transaction service exceptions

The exceptions that the WebSphere Application Server transaction service can throw are listed with a summary of each exception.

The exceptions are grouped as follows:

- Standard exceptions
- Heuristic exceptions

If the EJB container catches a system exception from the business method of an enterprise bean, and the method is running within a container-managed transaction, the container rolls back the transaction before passing the exception on to the client. For more information about how the container handles the exceptions thrown by the business methods for beans with container-managed transaction demarcation, see the section *Exception handling* in the Enterprise JavaBeans 2.0 specification. That section specifies the container action as a function of the condition under which the business method executes and the exception thrown by the business method. It also illustrates the exception that the client receives and how the client can recover from the exception.

Standard exceptions

The standard exceptions such as `TransactionRequiredException`, `TransactionRolledbackException`, and `InvalidTransactionException` are defined in the Java Transaction API (JTA) 1.1 specification.

InvalidTransactionException

This exception indicates that the request carried an invalid transaction context.

TransactionRequiredException exception

This exception indicates that a request carried a null transaction context, but the target object requires an active transaction.

TransactionRolledbackException exception

This exception indicates that the transaction associated with processing of the request has been rolled back, or marked for roll back. Thus the requested operation either could not be performed or was not performed because further computation on behalf of the transaction would be fruitless.

Heuristic exceptions

A heuristic decision is a unilateral decision made by one or more participants in a transaction to commit or rollback updates without first obtaining the consensus outcome determined by the Transaction Service. Heuristic decisions are an issue only after the participant has been prepared and the second phase of commit processing is underway. Heuristic decisions are typically made only in unusual circumstances, such as repeated failures by the transaction manager to communicate with a resource manager during two-phase commit. If a heuristic decision is taken, there is a risk that the decision differs from the consensus outcome, resulting in a loss of data integrity.

The following list provides a summary of the heuristic exceptions. For more detail, see the Java Transaction API (JTA) 1.1 specification.

HeuristicRollback exception

This exception is raised on the commit operation to report that a heuristic decision was made and that all relevant updates have been rolled back.

HeuristicMixed exception

This exception is raised on the commit operation to report that a heuristic decision was made and that some relevant updates have been committed and others have been rolled back.

Transaction exceptions that involve both one-phase and two-phase commit resources

The exceptions that can be thrown by transactions that involve one-phase and two-phase commit resources are the same as those that can be thrown by transactions involving only two-phase commit resources.

The exceptions that can be thrown are listed in the application programming interface (API) reference information in the WebSphere Application Server Information Center.

Chapter 24. Troubleshooting web applications

This page provides a starting point for finding information about web applications, which are comprised of one or more related files that you can manage as a unit, including:

- HTML files
- Servlets can support dynamic web page content, provide database access, serve multiple clients at one time, and filter data.
- Java ServerPages (JSP) files enable the separation of the HTML code from the business logic in web pages.

IBM extensions to the JSP specification make it easy for HTML authors to add the power of Java technology to web pages, without being experts in Java programming. More introduction...

Troubleshooting web applications

Web application deployment troubleshooting tips

Deployment of a web application is successful if you can access the application by typing a Uniform Resource Locator (URL) in a browser or if you can access the application by following a link. If you cannot access your application, follow these steps to eliminate some common errors that can occur during migration or deployment.

Web module migrated from version 4.x does not run in later WebSphere Application Server version.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Symptom	Your version 4.x web module does not run when you migrate it to Version 8.0 or later products.
Problem	In version 4.x, the classpath setting that affected visibility was <i>Module Visibility Mode</i> . In versions 6.0 and later, you must use class loader policies to set visibility.
Recommended response	Reassemble an existing module, or change the visibility settings in the class loader policies. Refer to the Class loaders and Class loading articles for more information.

Welcome page is not visible.

Symptom	You cannot access an application with a web path of: <code>/webapp/myapp</code>
Problem	The default welcome page for a web application is assumed to be <i>index.html</i> . You cannot access the default page of the <i>myapp</i> application unless it is named <i>index.html</i> .
Recommended response	To identify a different welcome page, modify the properties of the Web module during assembly, see the topic, Assembling web applications for more information in the <i>Developing and deploying applications</i> PDF book.

HTML files are not found.

Symptom	Your web application ran successfully on prior versions, but now you encounter errors that the welcome page (typically <i>index.html</i>), or referenced HTML files are not found: Error 404: File not found: Banner.html Error 404: File not found: HomeContent.html
Problem	For security and consistency reasons, web application URLs are now case-sensitive on all operating systems. Suppose the content of the index page is as follows: <pre><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 5.0 Frameset//EN"> <HTML> <TITLE> Insurance Home Page </TITLE> <frameset rows="18,80"> <frame src="Banner.html" name="BannerFrame" SCROLLING=NO> <frame src="HomeContent.html" name="HomeContentFrame"> </frameset> </HTML></pre> However the actual file names in the <code>\WebSphere\AppServer\installedApps\...</code> directory where the application is deployed are: banner.html homecontent.html
Recommended response	To correct this problem, modify the <i>index.html</i> file to change the names <i>Banner.html</i> and <i>HomeContent.html</i> to <i>banner.html</i> and <i>homecontent.html</i> to match the names of the files in the deployed application.

Proxy server cannot access a Web module

If you use the same context root when you install two applications that have the same Web module, and one of the applications is disabled, you are not able to use a proxy server to access the Web module. When this situation occurs, a 503 Service Unavailable error message is recorded in the SystemOut and SystemErr logs.

bprac: You should use a different context root for the Web module in each application, or use an application server instead of a proxy server to access the Web module.

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.

JavaServer Pages troubleshooting tips

Use these tips to troubleshoot problems with JavaServer Pages.

JavaServer Pages source code shown by the web server

If you share the document root of the WebSphere Application Server with the web server document root, a security exposure can result as the web server might display the JavaServer Pages (JSP) source file as plain text.

Problem

You can use the WebSphere Web server plug-in set of rules to determine whether a given request will be handled by the WebSphere Application Server. When an incoming request fails to match those rules, the web server plug-in returns control to the web server so that the web server can

fulfill the request. In this case, the unknown host header causes the web server plug-in to return control to the web server because the rules do not indicate that the WebSphere Application Server should handle it. Therefore, the web server looks for the request in the web server document root. Since the JSP source file is stored in the document root of the web server, the web server finds the file and displays it as plain text.

Suggested solution

Move the WebSphere Application Server JSP source file outside of the web server document root. Then, when this request comes in with the unknown host header, the plug-in returns control to the web server and the JSP source file is not found in the document root. Therefore, the web server returns a 404 File Not Found error rather than the JSP source file.

Problems displaying double-byte character set (DBCS) characters when using the @include directive

JavaServer Pages files that use the @include directive might experience problems when displaying double-byte character set (DBCS) characters. Some applications that are migrated to WebSphere Application Server Version 6.0 and above might need to be modified to comply with the JSP 2.0 specification as a result of backwards compatibility issues. The JSP 2.0 specification requires that each statically included resource must set a page encoding or content type because the character encoding for each file is determined separately, even if one file includes another using the include directive.

Problems using the JavaServer Pages (JSP) engine

If you are having difficulty using the JavaServer Pages (JSP) engine, try these steps:

1. Determine whether other resources such as .html files or servlets are being requested and displayed correctly. If they are not, the problem probably lies at a deeper level, such as with the HTTP server.
2. If other resources are being displayed correctly, determine whether the JSP processor has started normally:
 - Browse the logs of the server hosting the JSP files you are trying to access. The following messages indicate that the JSP processor has started normally:

```
Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
was initialized successfully.
Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
has been associated with patterns [*.jsp *.jspx *.jsw *.jsw ].
```

If the JSP processor fails to load, you will see a message such as

```
No Extension Processor found for handling JSPs.
JSP Processor not defined. Skipping : jspfilename.
```

in the server log files.

3. If the JSP engine has started normally, the problem may be with the JSP file itself.
 - The JSP may have invalid JSP syntax and could not be processed by the JSP Processor. Examine the server log files of the target application for invalid JSP directive syntax messages. Errors similar to the following in a browser indicate this kind of problem:

```
Message: /filename.jsp(2,1)JSPG0076E: Missing required attribute page for jsp
element jsp:include
```

This example indicates that line 2, column 1 of the named JavaServer Pages file is missing a mandatory attribute for the jsp:include action. Similar messages are displayed for other syntax errors.

- Examine the target application server's SystemErr.log files for problems with invalid Java syntax. Errors similar to Message: Unable to compile class for JSP in a browser indicate this kind of problem.

The error message output from the Javac compiler will be found in the SystemErr.log file. It might look like:

```
JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
int myInt=121;
    ^ 1 error
```

Correct the error in the JSP file and retry the file.

- Examine the log files for the target application for problems with invalid Java syntax. Errors similar to Message: Unable to compile class for JSP in a browser indicate this kind of problem.

The error message output from the Javac compiler will be found in the server log files. It might look like:

```
JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
int myInt=121;
    ^ 1 error
```

Correct the error in the JSP file and retry the file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log, SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

JavaServer Pages fail to compile when using precompile

Symptom JavaServer Pages fail to compile during deployment through the administrative console when precompile is selected.

```
SystemErr R com.ibm.websphere.management.exception.AdminException:
ADMA0021E: Error in compiling jsps - xyz.war (rc=1)
```

Problem JavaServer Pages fail to compile during deployment through the administrative console when precompile is selected when there is a dependency on another Java archive (JAR) file that is not available on any class path.

Suggested solution You may use wsadmin scripting to precompile JSP files during enterprise application deployment. However if you want to use the administrative console, then compile all JSP files before packaging the application.

1. Add the dependent JAR to the deployment manager in a cell environment.
 - a. Click **System Administration > Deployment manager > Java and Process Management > Process Definition > Java Virtual Machine** in the console navigation.
 - b. Add fully qualified dependent JAR in class path field.
 - c. Click OK.
 - d. Restart deployment manager.

JSPG0089E: Mismatch found between page directive encoding Shift_JIS and xml prolog encoding UTF-8

Symptom	The following error appears: JSP Processing Error HTTP Error Code: 500 Error Message: /test.jsp(2,1) /test.jsp(2,1) JSPG0089E: Mismatch found between page directive encoding Shift_JIS and xml prolog encoding UTF-8
Problem	The pageEncoding attribute in the jsp:directive.page element is not UTF-8.
Suggested solution	JavaServer Pages must specify a prolog that matches the encoding specified in the page directive. For example, <pre><?xml version="1.0" encoding="Shift_JIS"?> <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"> <jsp:directive.page language="java" contentType="text/html"; charset=Shift_JIS" pageEncoding="Shift_JIS"/> <jsp:text>XXXXXjsp:text>XXXXX> </jsp:root></pre> For additional information, see section JSP.4.1, Page Character Encoding, in the JavaServer Pages specification and section 4.3.3 and appendix F.1 of the Extensible Markup Language (XML) specification

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in the topic, 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. The IBM Support page contains documents that can save you time gathering information needed to resolve this problem.

Troubleshooting contexts and dependency injection

Use this task to troubleshoot errors related to Contexts and Dependency Injection (CDI) for the Java Platform, Enterprise Edition (Java EE) applications.

About this task

When you use a CDI implementation, you might experience errors during application deployment or when CDI interacts with other Java EE components. You might also experience problems with producers, interceptors and decorators or diagnostic trace. Use this task to fix these errors that might occur.

Procedure

- Troubleshoot application deployment problems. At application deployment time, the container must validate each injection point in the application and is satisfied by only one source of that dependency.

1. Resolve ambiguous dependency.

An ambiguous dependency, or `javax.enterprise.inject.AmbiguousResolutionException` exception, occurs when the container resolves an injection points type and qualifiers to more than one managed bean, producer method, or producer field.

The exception tells you the injection point that was being processed, and the candidate beans that remained at the end of the type-safe resolution process. Anything other than a single bean is an application error:

```
javax.enterprise.inject.AmbiguousResolutionException:
```

```
There is more than one api type with : com.ibm.websphere.samples.AppObject
with qualifiers : Qualifiers: [@com.ibm.websphere.samples.MyQualifier()]
```

```
for injection into Field Injection Point, field name : loginCheck,
  Bean Owner : [Name:loginBean,WebBeans Type:MANAGED,API Types:[java.lang.
Object,com.ibm.websphere.samples.LoginBean],
  Qualifiers:[javax.enterprise.inject.Any,javax.enterprise.inject.
Default,javax.inject.Named]]
```

```
found beans: Name:newObject,WebBeans Type:PRODUCERMETHOD, API Types:[com.ibm.websphere.
samples.AppObject,java.lang.Object], Qualifiers:[javax.enterprise.inject.
Any,com.ibm.websphere.samples.MyQualifier,javax.inject.Named]
```

```
Name:newObject2,WebBeans Type:PRODUCERMETHOD, API Types:[com.ibm.websphere.
samples.AppObject,java.lang.Object], Qualifiers:[javax.enterprise.inject.
Any,com.ibm.websphere.samples.MyQualifier,javax.inject.Named]
```

To resolve the error complete one of the following actions:

- Disambiguate the injection point by adding a qualifier to the injection point and if necessary, one source of the dependency.
- If necessary, annotate one of the sources of contextual instances with `@Alternative` or `@Specializes`, if some of the dependencies should not be considered for injection.

2. Resolve an unsatisfiable dependency.

An unsatisfiable dependency, or `javax.enterprise.inject.UnsatisfiedResolutionException`, occurs when there is no corresponding source for objects matching an injection point in the application. The API type of the field, along with the optional set of qualifier annotations, dictates the set of beans that are valid to satisfy the dependency. Causes of unsatisfiable dependency are as follows:

- There is no managed bean that is assignable to the type on the injection point.
- There is no producer method of any managed bean whose return type is assignable to the injection point.
- There is no producer field in any managed bean whose type is assignable to the injection point.
- One of the previously mentioned scenarios are valid, but the Qualifier annotations on the injection point are not present on the bean or producer.

Resolve the error by making a dependency with the API type and qualifiers available by introducing a new bean, removing qualifiers, or adding producers fields or methods. Section 5.2 of the Contexts and Dependency Injection for Java specification describes the type-safe resolution in detail.

```
javax.enterprise.inject.UnsatisfiedResolutionException: Api type
[com.ibm.websphere.samples.myType] is not found with the qualifiers
  Qualifiers: [@javax.enterprise.inject.Default()]
for injection into Field Injection Point, field name : loginCheck,
  Bean Owner : [Name:loginBean,WebBeans Type:MANAGED,
  API Types:[java.lang.Object,com.ibm.websphere.samples.LoginBean],
  Qualifiers:[javax.enterprise.inject.Any,javax.enterprise.
inject.Default,javax.inject.Named]]
```

3. Resolve passivating scope dependencies.

Passivation is the act of moving an idle object that is held in memory auxiliary storage. A passivating scope, such as the built-in scopes, `@SessionScoped` and `@ConversationScoped`, requires that any bean using the scope be passivation-capable. A bean is passivation-capable if it is either a stateful session bean or any other managed bean that is both serializable and has no non-serializable interceptors and decorators. Causes of passivating scope dependencies include the following:

- Changing the scope of an existing bean to a passivating scope, such as `@SessionScoped` or `@ConversationScoped`.
- Adding non-serializable decorators or interceptors to an existing passivation capable bean.

Resolve the error as follows:

- Ensure that the bean in question is serializable.
- Ensure all interceptors and decorators of the bean implement serializable.
- Change the scope of the managed bean to a non-passivating scope.

In the following exception, the myCDIBean bean has java.io.Serializable as an API type, therefore, the problem is an interceptor or decorator of this bean:

```
Caused by: org.apache.webbeans.exception.WebBeansConfigurationException:
Passivation scoped defined bean must be passivation
capable, but bean : Name:myCDIBean,WebBeans Type:MANAGED,API
Types:[com.ibm.websphere.samples.myCDIBean java.io.Serializable,java.lang.
Object,com.ibm.websphere.samples.myLocalIface],
Qualifiers:[javax.enterprise.inject.Any,javax.enterprise.inject.
Default,javax.inject.Named] is not passivation capable
```

- Troubleshoot errors that result from CDI interacting with other Java EE components.

The @Inject annotation provides an additional type of Java EE dependency injection. Its relation to injection that is defined in Java EE 5 is as follows:

- Injection using annotations other than the @Inject annotation behave as in previous releases, and only dependencies injected using the @Inject annotation are contextual instances as defined by Contexts and Dependency Injection for Java (JSR299).
- Use producer fields and producer methods to provide limited CDI features (such as type-safe injection) of Java EE dependencies obtained using the @Resource, @PersistenceContext, @PersistenceUnit, and @WebServiceRef annotations.

If you cannot obtain a value for an Expression Language (EL) reference to a managed bean for JavaServer Pages (JSP) and JavaServer Faces (JSF) components, consider the following approaches:

- Ensure that the bean class is annotated using the @Named annotation or is annotated with a stereotype that defines the @Named annotation.
- Ensure that the EL expression matches the class name of the bean class, after converting the first character to lowercase. When you use the @Named annotation qualifier with a value member (for example, @Named("myName")), this specifies the bean name (a special case qualifier) but does not change the EL name of that bean.

For EJB components:

- You can inject session beans with the @Inject and @EJB annotations. When you inject stateful session beans with the @Inject annotation, the session beans can take advantage of type-safe injection using qualifiers, and can have their life cycle managed by their CDI scope.
- Session beans are eligible for interception and decoration even when they are not obtained with the @Inject annotation, unlike other managed beans.

For web service components:

- To develop a JAX-WS client from a Web Services Description Language (WSDL) file, follow the steps outlined in the topic, Developing a JAX-WS client from a WSDL file.

Tip: Use the wsimport tool to generate portable Java artifacts, including a service class.

- If you want to inject the generated service class into a CDI-managed bean, using the @WebServiceRef annotation, you must invoke the wsimport tool using the -wsdllocation argument. As a result, the generated service class is portable to other systems because the service class references the WSDL file using a relative URI, instead of an absolute path.

- Troubleshoot producer errors.

1. Looping in producer methods. When you use a producer method, each parameter is treated as an injection point in which the container provides the dependency. Therefore, the source of contextual objects that fulfill those parameter injection points must not be the same class that contains the producer method.
2. Duplicate producer methods (two @Produces annotations with same qualifiers in the same class). If a class has multiple producer fields, these fields cannot have the same API type and an identical set of qualifiers, as such a guaranteed ambiguous dependency would not be injectable.

```
org.apache.webbeans.exception.definition.DuplicateDefinitionException:
PassivationCapable bean id is not unique:
PRODUCERFIELD#class#com.ibm.websphere.samples.AppObject#
@javax.enterprise.inject.Any(),@com.ibm.websphere.samples.AppScopeBinding2
```

```
( ),@javax.inject.Named(value=),
    bean:Name:a2b,WebBeans Type:PRODUCERFIELD,API Types:[java.lang.Object,
com.ibm.websphere.samples.AppObject],
    Qualifiers:[javax.enterprise.inject.Any,com.ibm.websphere.samples
.AppScopeBinding2,javax.inject.Named]
```

- Troubleshoot interceptor and decorator errors.
 1. Enable interceptor, decorator enablement interceptors, and decorators in the beans.xml file. All except EJB session beans apply only to contextual instances of beans. Contextual instances are instances obtained using the @Inject annotation or by calling methods on the BeanManager interface.
 2. Interceptors and decorators in multiple bean deployment archives (BDA). The set of enabled interceptors and decorators of a bean class are a collection of the enabled interceptors and decorators across the entire EAR file. The order of interceptors and decorators that are defined in different beans.xml files is undefined.
- Use diagnostic trace to help determine why an error occurred.
 1. Obtain a trace for CDI by specifying JCDI=all:com.ibm.ws.webbeans*=all:org.apache.webbeans*=all.
 - Obtain a list of all discovered managed beans by searching for org.apache.webbeans.config.BeansDeployer in trace.log file. Each managed bean, along with its type (interceptor, decorator, producer, enterprise (EJB)) is displayed.
 - Obtain a detailed listing of each bean and its type and qualifiers by searching for the getBeans string.
 2. Obtain an additional trace for interacting with Java EE injection and the EJB container by specifying EJBContainer=all:MetaData=all:Injection=all.
 3. Obtain an additional trace for interacting with web-related scopes and life cycles by specifying all:com.ibm.ws.wswebcontainer*=all.
- Avoid heavyweight operations in default constructors of managed beans.

Each injection point for a given managed bean receives a new client proxy that calls the default constructor of the underlying bean class, in addition to the actual bean instance that might be created when using the proxy. Additionally, because dependency injections occur after the constructor completes, constructors cannot use injected dependencies. See the @PostConstruct annotation life cycle callback for a place to put post-injection logic that runs in the underlying instance only.

Troubleshooting HTTP sessions

HTTP session manager troubleshooting tips

Use troubleshooting tips for problems creating or using HTTP sessions with your web application hosted by WebSphere Application Server.

Here are some steps to take:

- View the logs for the application server which hosts the problem application:
 1. Look at the messages that are written while each application is starting. Specifically, see the messages that are written between the following two messages:


```
Starting application: application
.....
Application started: application
```
 2. Within these messages related to starting the application, look for any errors or exceptions containing a package name of com.ibm.ws.webcontainer.httpsession. If none are found, this is an indication that the session manager started successfully.
 3. The error message, **SRVE0054E: An error occurred while loading session context and web application**, indicates that SessionManager did not start properly for a given application.
 4. Look within the logs for any messages related to the Session Manager. These messages are in the format SESNxxxxE for error messages and SESNxxxxW for warning messages, and xxxx specifies the number for the error. Look up the extended error definitions in the Session Manager message table.

-
- See the Best practices for using HTTP Sessions section in the *Developing and deploying applications* PDF books for more details.
- Alternatively, a special servlet can be invoked that displays the current configuration and statistics related to session tracking.
 - Servlet name: **com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
 - It can be invoked from any web module that is enabled to serve by class name. For example, using default_app, **http://localhost:9080/servlet/com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
 - If you are viewing the module using the serve-by-class-name feature, be aware that this module might be viewable by anyone who can view the application. You can map a specific, secured URL to the servlet instead and disable the serve-servlets-by-classname feature.
- If you are using **database-based persistent sessions**, look for problems related to the **data source** the Session Manager relies on to keep session state information. For details on diagnosing database related problems see the Errors accessing a data source or connection pool in the *Administering applications and their environment* PDF book

Error message SRVE0079E Servlet host not found after you define a port

Error message SRVE0079E can occur after you define the port in WebContainer > HTTP Transports for a server, indicating that you do not have the port defined in your virtual host definitions. To define the port,

1. On the administrative console, go to Environment > Virtual Hosts > default_host> Host Aliases> New
2. Define the new port on host "*"

The application server gets EC3 - 04130007 ABENDs

To prevent an EC3 - 04130007 abend from occurring on the application server, change the HTTP Output timeout value. The custom property *ConnectionResponseTimeout* specifies the maximum number of seconds the HTTP port for an individual server can wait when trying to read or write data. For instructions on how to set *ConnectionResponseTimeout*, see HTTP transport custom properties section of the *Administering applications and their environment* PDF book.

If these steps do not address your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes). If you do not find your problem referenced on this site, contact IBM support.

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.

HTTP session problems

This article provides troubleshooting information related to creating or using Hypertext Transfer Protocol (HTTP) sessions.

To view and update the session manager settings discussed here, use the administrative console. Select the application server that hosts the problem application, then under **Additional properties**, select **Web Container**, then **Session manager**.

What kind of problem are you having?

- “HTTP sessions are not getting created, or are lost between requests” on page 228
- “HTTP Sessions are not persistent ” on page 229
- “Session is shared across multiple browsers on same client machine ” on page 229
- “Session is not getting invalidated immediately after specified session timeout interval” on page 229

- “Unwanted sessions are being created by JavaServer Pages” on page 229
- “HTTP session problems” on page 227
- “Users are not logged out after the HTTP session timer expires” on page 230
- Exceptions can occur when updating applications with session persistence enabled during run time

HTTP sessions are not getting created, or are lost between requests

By default, the session manager uses cookies to store the session ID on the client between requests. Unless you intend to avoid cookie-based session tracking, ensure that cookies are flowing between WebSphere Application Server and the browser:

- Make sure the **Enable cookies** check box is checked under the **Session tracking Mechanism** property.
- Make sure cookies are enabled on the browser you are testing from or from which your users are accessing the application.
- Check the Cookie domain specified on the SessionManager (to view the or update the cookie settings, in the **Session tracking mechanism->enable cookies** property, click **Modify**).
 - For example, if the cookie domain is set as ".myCom.com", resources should be accessed using that domain name. Example: `http://www.myCom.com/myapp/servlet/sessionServlet`.
 - If the domain property is set, make sure it begins with a dot (.). Certain versions of Netscape do not accept cookies if domain name doesn't start with a dot. Internet Explorer honors the domain with or without a dot. For example, if the domain name is set to `mycom.com`, change it to `.mycom.com` so that both Netscape and Internet Explorer honor the cookie.

Note: When the servers are on different hosts, ensure that session cookies flow to all the servers by configuring a front-end router such as a web server with the plug-in or setting the Cookie domain.

- Check the **Cookie path** specified on the SessionManager. Check whether the problem URL is hierarchically below the Cookie path specified. If not correct the Cookie path.
- If the Cookie maximum age property is set, ensure that the client (browser) machine's date and time is the same as the server's, including the time zone. If the client and the server time difference is over the "Cookie maximum age" then every access would be a new session, since the cookie expires after the access.
- If you have multiple web modules within an enterprise application that track sessions:
 - If you want to have different session settings among web modules in an enterprise application, ensure that each web module specifies a different cookie name or path, or
 - If web modules within an enterprise application use a common cookie name and path, ensure that the HTTP session settings, such as Cookie maximum age, are the same for all web modules. Otherwise cookie behavior is unpredictable, and depends upon which application creates the session. Note that this does not affect session data, which is maintained separately by the web module.
- Check the cookie flow between browser and server:
 1. On the browser, enable "cookie prompt". Hit the servlet and make sure cookie is being prompted.
 2. Access the session servlet from the browser.
 3. The browser prompts for the cookie; note the `jsessionid`.
 4. Reload the servlet, note down the cookie if a new cookie is sent.
 5. Check the session trace and look for the session ID and trace the request by the thread. Verify that the session is stable across web requests:
 - Look for **getHttpSession(...)** which is start of session request.
 - Look for **releaseSession(..)** which is end of servlet request.
- If you are using URL rewriting instead of cookies:
 - Ensure there are no static HTML pages on your application's navigation path.
-

Note: Session tracking using the SSL ID is deprecated in WebSphere Application Server version 7.0. You can configure session tracking to use cookies or modify the application to use URL rewriting. If you are using SSL as your session tracking mechanism:

- Ensure that you have SSL enabled on your IBM HTTP Server or iPlanet HTTP server.
- If you are in a clustered (multiple node) environment, ensure that you have session persistence enabled.

HTTP Sessions are not persistent

If your HTTP sessions are not persistent, that is session data is lost when the application server restarts or is not shared across the cluster:

- Check the data source.
- Check the session manager's persistence settings properties:
 - If you intend to take advantage of session persistence, verify that Persistence is set to **Database**.
 - If you are using **Database-based persistence**:
 - Check the JNDI name of the data source specified correctly on SessionManager.
 - Specify correct userid and password for accessing the database.

Note that these settings have to be checked against the properties of an existing data source in the administrative console. The session manager does not automatically create a session database for you.

 - The data source should be non-JTA, for example, non XA enabled.
 - Check the logs for appropriate database error messages.
 - With DB2, for row sizes other than 4k make sure specified row size matches the DB2 page size. Make sure tablespace name is specified correctly.
 - If you are using **memory-based persistence** (available only in a network deployment environment):
 - Review the Memory-to-memory replication section of the *Developing and deploying applications* PDF book.
 - Review the **Internal Replication Domains properties** of your session manager.

Session is shared across multiple browsers on same client machine

This behavior is browser-dependent. It varies between browser vendors, and also may change according to whether a browser is launched as a new process or as a subprocess of an existing browser session (for example by hitting Ctl-N on Windows).

The Cookie maximum age property of the session manager also affects this behavior, if cookies are used as the session-tracking mechanism. If the maximum age is set to some positive value, all browser instances share the cookies, which are persisted to file on the client for the specified maximum age time.

Session is not getting invalidated immediately after specified session timeout interval

The SessionManager invalidation process thread runs every x seconds to invalidate any invalid sessions, where x is determined based on the session timeout interval specified in the session manager properties. For the default value of 30 minutes, x is around 300 seconds. In this case, it could take up to 5 minutes (300 seconds) beyond the timeout threshold of 30 minutes for a particular session to become invalidated.

Unwanted sessions are being created by JavaServer Pages

As required by the JavaServer Pages (JSP) specification, JSP pages by default perform a `request.getSession(true)`, so that a session is created if none exists for the client. To prevent JSP pages from creating a new session, set the session scope to **false** in the `.jsp` file using the page directive as follows:

```
<% @page session="false" %>
```

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.

Unexpected re-authentications: When you set the `com.ibm.ws.security.web.logoutOnHTTPSessionExpire` custom property to `true`, unexpected re-authentications might occur when you are working with multiple web applications. By default, each web application has its own unique HTTP session, but the web browser has one session cookie. To address this issue, you can change the HTTP session configuration by giving each application a unique session cookie name or path setting. As a result, each application gets its own session cookie. Alternatively, you can configure multiple web applications with the same enterprise application to share the same HTTP session. For more information, see the [Assembling so that session data can be shared](#) topic.

Exceptions can occur when updating applications with session persistence enabled during run time

Users who have session persistence enabled and execute application updates during run time may experience unexpected exceptions after the application is restarted.

If updates have been made that change the attributes saved, then all the sessions created by the associated application might have to be invalidated prior to the application update if the application can not handle these changes. In this situation, all session objects must be removed from the back-end as well. Read the session management invalidation chapters for further details on how to remove these sessions properly.

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/software/webservers/appserv/zos_os390/support/

Chapter 25. Troubleshooting web services

This page provides a starting point for finding information about web services.

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. They implement a services oriented architecture (SOA), which supports the connecting or sharing of resources and data in a very flexible and standardized manner. Services are described and organized to support their dynamic, automated discovery and reuse.

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:

Procedure

- 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.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - z/OS

app_server_root

Refers to the top directory for a WebSphere Application Server node.

The node may be of any type—application server, deployment manager, or unmanaged for example. Each node has its own *app_server_root*. Corresponding product variables are *was.install.root* and *WAS_HOME*.

The default varies based on node type. Common defaults are *configuration_root/AppServer* and *configuration_root/DeploymentManager*.

configuration_root

Refers to the mount point for the configuration file system (formerly, the configuration HFS) in WebSphere Application Server for z/OS.

The *configuration_root* contains the various *app_server_root* directories and certain symbolic links associated with them. Each different node type under the *configuration_root* requires its own cataloged procedures under z/OS.

The default is */wasv8config/cell_name/node_name*.

plug-ins_root

Refers to the installation root directory for Web Server Plug-ins.

profile_root

Refers to the home directory for a particular instantiated WebSphere Application Server profile.

Corresponding product variables are *server.root* and *user.install.root*.

In general, this is the same as *app_server_root/profiles/profile_name*. On z/OS, this will always be *app_server_root/profiles/default* because only the profile name "default" is used in WebSphere Application Server for z/OS.

smpe_root

Refers to the root directory for product code installed with SMP/E or IBM Installation Manager.

The corresponding product variable is *smpe.install.root*.

The default is */usr/lpp/zWebSphere/V8R0*.

Web services command-line tools troubleshooting tips

Use these tips to troubleshoot 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 WSDL2Java and Java2WSDL command-line tools are not supported on the z/OS platform. This functionality is provided by the assembly tools provided with WebSphere Application Server running on the z/OS platform. Read about the WSDL2Java and Java2WSDL command-line tools for Java API for XML-based Remote Procedure Call (JAX-RPC) applications to learn more about these tools.

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 WebSvcInSession20EJB.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>
    <wsdlport>
        <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>
</webservises>

```

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.WSMultiProtocolWebSvcBean</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.bat file when using a Windows operating system or the WSDL2Java.sh file if you are using a Linux or AIX operating system. These files are located in the <install_root>/WebSphere/AppServer/bin directory.

If you use a Windows operating system, set your proxy host and port values in the WSDL2Java.bat file as follows:

```

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

```

If you use a Linux or AIX operating system, set your proxy host and port values in the WSDL2Java.sh file as follows:

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

Emitter failure error occurs when running the WSDL2Java command on a WSDL document containing a JMS-style endpoint URL

If you run the WSDL2Java command-line tool on a WSDL document that contains a JMS-style endpoint web address, for example, `jms:/...`, the `urlprotocols.jar` file that contains the custom protocol handler for the JMS protocol must be in the CLASSPATH variable statement. The error **WSWS3099E: Error: Emitter failure. Invalid endpoint address in port <x> in service <y>: <jms-url-string>** is avoided by making sure the `urlprotocols.jar` file is in the CLASSPATH variable statement.

To add the `urlprotocols.jar` file to the CLASSPATH variable statement:

On Windows platforms, edit the `install_root\bin\setupCmdLine.bat` file and locate the line that sets the `WAS_CLASSPATH` environment variable. Add `%install_root%\lib\urlprotocols.jar` to the end of the line that sets the `WAS_CLASSPATH` environment variable.

On Linux, AIX, HP-UX, and Solaris operating systems, edit the `install_root/bin/setupCmdLine.sh` file and add `$install_root/lib/urlprotocols.jar` to the end of the line that sets the `WAS_CLASSPATH` environment variable.

Make sure to use the proper delimitator character for your platform, for example, use a semicolon (;) for Windows platforms and a colon (:) for Linux, AIX, HP-UX, and Solaris operating systems.

Web services compiled bindings troubleshooting tips

Use these tips to troubleshoot 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 Services 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 `-introspect` 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, WWS3422E: 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, WWS3713E: 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 configuring web services client bindings information to learn how to configure the port information.

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 ConnectionLOutTimeout 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.

To solve the problem, increase the `syncTimeout` parameter for the web services client. To learn how to set this parameter, see the configuring the JAX-RPC web services client bindings in the `ibm-webservicesclient-bnd.xmi` deployment descriptor information.

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

Use these tips to troubleshoot 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.addElement(ExtWSPolicyData_Ser.java: 210)
at com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.serialize(ExtWSPolicyData_Ser.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.Calendar getDate() throws java.rmi.RemoteException;
}
```

The client version of the `server.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 the `NoSuchMethod` error, the `IncompatibleClassChangeError` and `ClassCastException` can occur. However, almost any runtime 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 these 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. There 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 specifies 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 website and download the latest fix pack for WebSphere Application Server, Version 5.0.2.

Application client SOAP request troubleshooting tips

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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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

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: `install_dir/installedApps/application_name.ear/soapsec.war/key/` directory. After replacing these files, you must stop and restart the application.

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 Web Services Invocation Framework (WSIF) troubleshooting tips information for messaging.
- Investigate the following areas for SOAP-related problems:
 - View the error log of the HTTP server to which the SOAP request is sent.
 - Browse the website <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 is documented in the diagnosing and fixing problems resources for learning information.

The IBM Support page provides current information available from IBM Support on known problems and their resolution. 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:

Procedure

1. Enable trace for a web services unmanaged client.
 - a. Create a trace properties file by copying the `%install_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=c:\\temp\\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:

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

To learn more about this tool, see the `launchClient` tool 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**.

To learn more about enabling trace, see the tracing and logging configuration information.

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 `makeWithdrawal` 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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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 runtime 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.

Procedure

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

```
java -cp app_server_root/runtimes/com.ibm.ws.webservices.thinclient_8.0.0.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 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 8.0 is based on Web Services for Java Platform, Enterprise Edition (Java EE) 6 and 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 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 the specifications and API information.

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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

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
```

Type the previous three lines as one continuous line.

Errors when securing web services

The following errors might occur when you secure web services:

- ““CWWSS6811E: The key identifier <identifier name> retrieved from the message is different from the key identifier <identifier name> acquired from the keystore" key mismatch error message displays” on page 251
- ““CWWSI5061E: The SOAP Body is not signed" error message displays” on page 251
- ““CWWSI5075E: No security token found that satisfies any one of the authentication methods" error message displays” on page 252
- ““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 253
- ““CWSCJ0053E: Authorization failed for /UNAUTHENTICATED..." error message displays” on page 253
- ““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 254
- ““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 254
- ““WSEC5502E: Unexpected element as the target element" error message displays ” on page 255
- ““WSEC6664E: Null is not allowed to PKIXBuilderParameters. The configuration of TrustAnchor and CertStoreList are not correct" exception displays” on page 255
- ““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 256
- ““WSEC6500E: There is no candidate used to login" error message displays” on page 259
- ““SHA-1 key identifier for Kerberos token is not consumed or generated correctly without a custom property” on page 260
- ““Kerberos V5 binary AP_REQ token is not generated or consumed correctly without a custom property” on page 261
- ““Instead of issuing a CertPath exception, a valid certification path is built on Sun Solaris when an invalid certificate is used” on page 261
- ““Hardware cryptographic requests with card-related exceptions must use cryptographic software to complete requests successfully” on page 262

"CWSS6811E: The key identifier <identifier name> retrieved from the message is different from the key identifier <identifier name> acquired from the keystore" key mismatch error message displays

Cause:

The sample keystore files included with the product have been updated because some of the certificates are due to expire. If the keys are used in a mixed cluster environment, a key mismatch error occurs. For example:

```
com.ibm.wsspi.wssecurity.core.SoapSecurityException: CWSS6521E: The Login failed because of an exception: javax.security.auth.login.LoginException: CWSS6811E: The key identifier TVpC640XSSc= retrieved from the message is different from the key identifier QdZLf+KjrUg= acqu
```

The keystore files included with the product are intended as samples and should not be used in a production environment. However, if you use the sample files to test a mixed cluster environment, the files should be synchronized to prevent the key mismatch error. The keystore files are located in a sub-directory under the profile directory. For example:

```
profile_root/etc/ws-security/samples/dsig-sender.ks
profile_root/etc/ws-security/samples/dsig-receiver.ks
profile_root/etc/ws-security/samples/enc-sender.jceks
profile_root/etc/ws-security/samples/enc-receiver.jceks
profile_root/etc/ws-security/samples/intca2.cer
```

where *profile_root* is the home directory for a particular instantiated WebSphere Application Server profile.

The sample keystore files are also located in the installation directory:

```
app_server_root/etc/ws-security/samples/dsig-sender.ks
app_server_root/etc/ws-security/samples/dsig-receiver.ks
app_server_root/etc/ws-security/samples/enc-sender.jceks
app_server_root/etc/ws-security/samples/enc-receiver.jceks
app_server_root/etc/ws-security/samples/intca2.cer
```

where *app_server_root* is the location of the WebSphere Application Server installation.

Solution:

You can work around the key mismatch error by manually copying the keystore files from the Version 7.0 and later nodes in the mixed cluster to the Version 6.1 Feature Pack for web services nodes. This ensures that Version 7.0 and later nodes and Version 6.1 Feature Pack for web services nodes are using the same keystore files. Another workaround is to move the Version 7.0 and later keystore files to a common directory location, then modify all bindings to point to the common location for the keystore files.

"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.

"CWSI5075E: 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, read about configuring the client for identity assertion when specifying the method and configuring the client for identity assertion collecting the authentication method.

To configure the server for identity assertion, read about configuring the server to handle identity assertion authentication and 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.1a
```

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

To configure the client for identity assertion, read about configuring the client for identity assertion when specifying the method and configuring the client for identity assertion collecting the authentication method.

"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 Services 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.

To learn more, see the assembly tools information. .

"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 about tracing web services.
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.

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.S 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.

"WSSecurityConsumer 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 WebServicesFault
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();
FredBankServiceLocator locator
    =(FredBankService)ctx.lookup("java:comp/env/service/FredBankService");
FredBank fb = locator.getFredBank(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);
FredBankServiceLocator locator = new FredBankServiceLocator();
FredBank fb = locator.getFredBank(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.S 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**.

SHA-1 key identifier for Kerberos token is not consumed or generated correctly without a custom property

Cause:

As specified in the OASIS standard titled Web Services Security Kerberos Token Profile v1.1, a base64 encoding of a SHA-1 transformed key value can be used to specify a key identifier for a Kerberos AP-REQ token. SHA-1 is a cryptographic hash function that transforms an input and returns a fixed size string. After the client and service provider have exchanged an initial web services message, the SHA-1 key identifier is used to externally reference the Kerberos authentication token. To use SHA-1 for this purpose, you must configure a custom property in the policy bindings to generate and consume the SHA-1 key identifier. The custom property `com.ibm.wsspi.wssecurity.kerberos.attach.hashKeySupportToken` is added to the client token generator and service token consumer bindings. This property allows the application to correctly generate and consume the SHA-1 key identifier during subsequent exchanges of web services messages when the Kerberos token is used as an authentication token.

Solution:

If an application generates or consumes a SHA-1 key identifier for each web services message exchange, set the `com.ibm.wsspi.wssecurity.kerberos.attach.hashKeySupportToken` custom property to `true` in the token generator and the token consumer bindings for the application.

To set the custom property using the administrative console, complete these steps:

1. Click **Services > Policy sets**.
2. Click **General provider policy set bindings > binding_name**.
3. Click the **WS-Security** policy in the Policies table.
4. Click **Authentication and protection** in the security policy bindings section.
5. Under **Authentication tokens**, click the name of the token consumer or token generator.

6. In the **Custom properties** section, enter the `com.ibm.wsspi.wssecurity.kerberos.attach.hashKeySupportToken` custom property and the property value of `true`.
7. Click **OK**, then click **Save**.

Note: You should consider the possibility of a man-in-the-middle attack which can intercept the SHA-1 key during message exchanges. To protect the SHA-1 key, use either transport-level security, such as SSL, or message-level security including signature and encryption.

Kerberos V5 binary AP_REQ token is not generated or consumed correctly without a custom property

Cause:

When Microsoft® web service applications request messages using a Kerberos token, you must configure a custom property in the policy bindings for the Kerberos V5 AP_REQ token to generate and consume the token. Add the `com.ibm.wsspi.wssecurity.kerberos.attach.apreq` custom property to the client token generator and service token consumer bindings. Enabling this property allows the application to generate and consume the Kerberos AP_REQ token for each web services request message.

Solution:

If an application generates or consumes a Kerberos V5 AP_REQ token for each web services request message, set the `com.ibm.wsspi.wssecurity.kerberos.attach.apreq` custom property to `true` in the token generator and the token consumer bindings for the application, as follows:

- To interoperate with Microsoft .NET client applications, set the custom property to `true` in the token consumer bindings for the target service.
- To interoperate with Microsoft .NET services, set the custom property to `true` in the client token generator bindings.
- If an application generates the Kerberos AP_REQ token for each web services message, set the custom property to `true` in both the client token generator bindings and the service token consumer bindings.

Note: Generating and consuming an AP_REQ token for every Web Services request message has implications for product performance, including message processing efficiency and memory usage.

To set this custom property in the administrative console, complete the following steps:

1. In the administrative console, click **Services > Policy sets**.
2. Click **General provider policy set bindings > binding_name**.
3. Click **WS-Security** policy in the Policies table.
4. Click **Authentication and protection** in the security policy bindings section.
5. Under Protection tokens, click the name of the token consumer or token generator.
6. In the Custom properties section, enter the `com.ibm.wsspi.wssecurity.kerberos.attach.apreq` custom property and the appropriate value (`true`).

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 IBM CertPath security provider is used. IBM CertPath 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 (IBM CertPath) 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
```

- When the web service is deployed to Sun Solaris, the `java.security.cert.CertPathBuilder.build` method returns the Sun default CertPath provider instead of IBM CertPath. 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 IBM CertPath 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:

```
CWSS5601E: The following exception occurred while decrypting the message:  
com.ibm.pkcs11.PKCS11Exception: Another operation is already active
```

and

```
CWSS5601E: 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.

Detecting and fixing problems with WS-ReliableMessaging

The nature of WS-ReliableMessaging is that network and server failures are assumed, and therefore the target web service or message store might not be available. In these cases, message sequences cannot be completed and collections of web service messages are held awaiting transmission. You can use the `SystemOut.log` file, system events, and the runtime administrative panels to monitor the system and detect and fix problems with WS-ReliableMessaging.

About this task

If a sequence fails, a message is written to the application server `SystemOut.log` file and a system event is generated. Therefore you can detect failed sequences by looking at the `SystemOut.log` file, or by writing an event listener (or by using third party software) to monitor system events.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.




Note: After a sequence has been established, WS-ReliableMessaging provides retransmission of messages to a service. However if the sequence is not established (that is, if the initial `CreateSequence` request is refused) then the messages are not transmitted to the service. For more information, see the troubleshooting tip “A sequence is not established, and therefore WS-ReliableMessaging cannot ensure messages are transmitted” on page 270.

For more detailed status information at run time, and facilities to help you fix problems, use the WS-ReliableMessaging administrative console runtime panels. These panels are available at many different scopes (for example cell; application server; messaging engine). For a full list of the WS-ReliableMessaging runtime panels, and details of the scopes at which they are available, see WS-ReliableMessaging - administrative console panels.

At all scopes, the parent panel is Reliable messaging state settings. From this panel you can investigate each of the three key runtime aspects of reliable messaging:

- Message stores
- Inbound sequences
- Outbound sequences

The following icons are displayed here and on several other reliable messaging runtime panels:

	OK	Everything here, and (if there is a link) in all runtime panels below this link, is running normally.
	Warning	Something here, or (if there is a link) in one of the runtime panels below this link, is in an unusual state and you might have to take some action to resolve it. For example, the system might be awaiting a response from an endpoint. In this case, either the response will be received (in which case you need take no action and the runtime information will be updated to "OK") or the reliable messaging destination has stopped acknowledging messages (in which case you have to take some action to resolve the failed sequence).
	Error	There is a definite error that you must take some action to resolve, either here or (if there is a link) in one of the runtime panels below this link.

Note that for troubleshooting purposes you only have to follow links to the sub-panels if states other than "OK" are displayed.

To use the reliable messaging runtime panels to detect and fix problems with WS-ReliableMessaging, complete one or more of the following steps:

Procedure

- Investigate problems with message stores.

In the navigation pane, click one of the paths to this panel. For example **Servers > Server Types > WebSphere application servers > *server_name* > [Additional Properties] Reliable messaging state > Runtime > Message store**. The list of reliable messaging storage managers for the current scope is displayed in the Message store collection form.

For the managed qualities of service, the messages are written to a messaging engine. For the unmanaged non-persistent quality of service, the messages are stored in memory. For in-memory stores the only possible value is "Running". For messages stored by a messaging engine, the possible values are "Running" or "Messaging engine not contactable", probably because the messaging engine is not running. The "OK" icon indicates that the message store is running. If the messaging engine is not contactable, the "Error" icon is displayed.

For each message store in the list, the name of the associated reliable messaging application is given in the *description* column. If a messaging engine is not contactable, restart the message store for that application.

- Investigate problems with inbound sequences.

In the navigation pane, click one of the paths to this panel. For example **Servers > Server Types > WebSphere application servers > *server_name* > [Additional Properties] Reliable messaging state > Runtime > Inbound sequences**. The runtime state of each of the inbound sequences for the current scope is displayed in the Inbound sequence collection form.

You can use a filter to look at sequences that are in a particular state (for example Failed due to missing message) or that have a large number of messages awaiting dispatch to applications. If the sequence status is Error, there is a problem with the sequence and the source server hosting the other end of the sequence has terminated it. If the sequence is active and there are a large number of messages awaiting dispatch to the application, then there might be a problem with the application or, if in-order delivery is specified, delivery might be held up because the sequence has gaps in it.

You can select one or more sequences, then use the buttons provided to dispatch the messages to their associated applications, to export the messages to compressed files, to close or terminate the selected sequences, or to delete the selected sequences and all their messages.

Attention: Delete or terminate sequences only if necessary. If you delete or terminate an active sequence, the resulting messaging behavior is unpredictable and can cause loss of messages. If you are not sure whether you can safely delete or terminate a sequence, do not delete or terminate it; the system automatically deletes sequences that have been inactive for 12 hours.

To see more detailed information about a particular sequence, click the Sequence identifier field. The Inbound sequences settings form is displayed. This detailed information includes addressing information to help you identify the source of the sequence, and the value (true or false) for **in-order delivery** for the sequence. From this panel you can also display the following forms:

- The Acknowledgement state collection form. (The ranges of message sequence numbers received from the WS-ReliableMessaging source. If more than one range is displayed, this indicates a gap in the messages received. If "In-order delivery" is selected for the sequence manager, messages with a sequence number greater than the lowest gap cannot be delivered to the application until the gap is closed.)
- The Inbound message collection form. (The messages on the inbound sequence. You can use this form to delete individual messages.)
 - The Message settings form. (The contents of an individual message in the sequence.)

For more guidance on diagnosing problems with inbound sequences, see "Diagnosing the problem when a reliable messaging source cannot deliver its messages" on page 266

- Investigate problems with outbound sequences.

In the navigation pane, click one of the paths to this panel. For example **Servers > Server Types > WebSphere application servers > *server_name* > [Additional Properties] Reliable messaging state > Runtime > Outbound sequences**. The runtime state of each of the outbound sequences for the current scope is displayed in the Outbound sequence collection form.

You can use a filter to look at sequences that are in a particular state. For example, the state Cannot contact the remote endpoint indicates that the sequence has been established but the reliable messaging destination has stopped acknowledging messages (which, coupled with a high number of messages awaiting transmission, might indicate a potential problem). If the sequence status is Error, there is a problem with the sequence and the server hosting the other end of the sequence has terminated it.

You can select one or more sequences, and use one of the buttons provided to export the messages to compressed files, to close or terminate the selected sequences, or to delete the selected sequences and all their messages. For more information about deleting sequences, see “Deleting a failed WS-ReliableMessaging outbound sequence” on page 267.

Attention: Delete or terminate sequences only if necessary. If you delete or terminate an active sequence, the resulting messaging behavior is unpredictable and can cause loss of messages. If you are not sure whether you can safely delete or terminate a sequence, do not delete or terminate it; the system automatically deletes sequences that have been inactive for 12 hours.

To see more detailed information about a particular sequence, click the Sequence identifier field. The Outbound sequences settings form is displayed. This detailed information includes addressing information to help you identify the server at which the sequence is targeted. From this panel you can also display the following forms:

- The Outbound message collection form. (The messages on the outbound sequence. You can use this form to delete individual messages.)
 - The Message settings form. (The contents of an individual message in the sequence.)

For more guidance on diagnosing problems with outbound sequences, see “Diagnosing and recovering a WS-ReliableMessaging outbound sequence that is in retransmitting state” on page 266.

WS-ReliableMessaging sequence reallocation

In some situations, the WS-ReliableMessaging implementation can recover from a sequence-related fault, so your application can continue without having to process the fault itself. Your application must still process the fault if the recovery fails.

When a server receives a request on a reliable messaging sequence that is no longer available for processing messages, a SOAP fault is produced. If the fault contains one of the following fault codes, and the message exchange pattern is asynchronous or synchronous one-way, the runtime environment creates a new sequence to the same endpoint and resends any messages that were due for delivery on the original sequence:

- wsrn:SequenceTerminated
- wsrn:MessageNumberRollover
- wsrn:UnknownSequence

These fault codes are contained in a wsrn:FaultCode element, within a wsrn:SequenceFault element in the SOAP header.

Any future messages to the target endpoint are also sent on the new sequence.

If the creation of the new sequence fails, the original fault is returned to the client. The client application must detect the fault and create a new sequence, by using the WS-ReliableMessaging system programming interfaces (SPIs), to resend the message.

If your application uses asynchronous messaging, responses from the provider to the client might also be reallocated in this way. Sequence reallocation does not occur when the message exchange pattern is synchronous two-way.

Note: Both the original sequence and the new sequence are visible in the administrative console panels. Do not delete the original sequence; it is automatically deleted after 12 hours. If you delete the original sequence while the new sequence is in use, messages can no longer be sent on the new sequence.

Diagnosing the problem when a reliable messaging source cannot deliver its messages

If you know the sequence identifier, and the target URI for the messages, you can use the runtime administrative console panels to examine the sequence and determine why a reliable messaging source is not delivering its messages into the application server.

About this task

To diagnose this problem, complete the following steps:

Procedure

1. Identify the target server or cluster from the target URI.
2. Use the administrative console to view the inbound sequences runtime information for the target server or cluster.

In the navigation pane, click either **Servers > Server Types > WebSphere application servers > *server_name* > [Additional Properties] Reliable messaging state > Runtime > Inbound sequences** or **Servers > Clusters > WebSphere application server clusters > *cluster_name* > [Additional Properties] Reliable messaging state > Runtime > Inbound sequences**. The runtime state of each of the inbound sequences for the current scope is displayed in the Inbound sequence collection form.

3. Specify a filter on the runtime view of the sequences, filtering on the Sequence identifier provided by the reliable messaging source owner.
4. Click on the Sequence identifier of the displayed sequence to get a more detailed view of it.
5. If the sequence has failed, examine the runtime status of the failed sequence and provide the reliable messaging source owner with details of which messages have been acknowledged, so that the owner can decide whether to delete, or to process out of order, any undelivered messages.
6. If the sequence has not failed, see whether there is a communications problem between the reliable messaging source and the application server.

Diagnosing and recovering a WS-ReliableMessaging outbound sequence that is in retransmitting state

You have to resolve a sequence in retransmitting state, because messages are backing up awaiting delivery to the target service. The retransmission might be due to a network failure, or a failure of the target server. The problem should resolve automatically, but you might want to investigate the cause to speed up recovery.

About this task

To resolve a sequence in retransmitting state, complete the following steps:

Procedure

1. Check for network failures.

2. If no network failures are found, look up the target endpoint address (URI) for the sequence to determine the owner of the target service, then contact the owner to check if the target server is available.

The target endpoint address is shown in the Outbound sequences settings form. Use the administrative console to navigate to this form, as described in “Detecting and fixing problems with WS-ReliableMessaging” on page 263.

3. After the communication link between the two servers is re-established, use the runtime panels to confirm that messages in the sequence are now being delivered.

Deleting a failed WS-ReliableMessaging outbound sequence

You have to resolve an outbound sequence in failed state, so that messages can again be transmitted to the target service. A sequence in failed state shows an unrecoverable error. The sequence can no longer be used. If messages are being delivered in order, then the failed sequence must be resolved before a new sequence can be established.

About this task

Deleting an outbound sequence allows the runtime environment to automatically create a new sequence the next time that an application attempts to invoke a web service at the destination address that the failed sequence was targeting. To work with outbound sequences you use the administrative console runtime panels as described in “Detecting and fixing problems with WS-ReliableMessaging” on page 263.

Attention: Delete or terminate sequences only if necessary. If you delete or terminate an active sequence, the resulting messaging behavior is unpredictable and can cause loss of messages. If you are not sure whether you can safely delete or terminate a sequence, do not delete or terminate it; the system automatically deletes sequences that have been inactive for 12 hours.

To diagnose and delete a failed outbound sequence, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane of the administrative console, click one of the paths to the outbound sequences collection form. For example **Servers > Server Types > WebSphere application servers > *server_name* > [Additional Properties] Reliable messaging state > Runtime > Outbound sequences**. The runtime state of each of the outbound sequences for the current scope is displayed in the Outbound sequence collection form.
2. Examine the failure reason by clicking on the Sequence identifier field of the failed sequence. The Outbound sequences settings form is displayed. The failure reason is based on the fault message received by the sequence manager from the target server.
3. If there are messages associated with the failed sequence, decide what to do with these messages. The messages might have been transmitted and received at the target server, or they might not. You might choose to delete messages from the sequence or export them to a compressed file. If you choose to delete the messages, you can either delete individual messages or you can delete all the messages.
 - a. Optional: To delete one or more messages from a failed sequence, complete the following steps:
 - 1) In the main pane of the Outbound sequences settings form, under the Additional Properties section, click **Messages**. The messages for the failed outbound sequence are listed in the Outbound message collection form.
 - 2) Select the check boxes next to the names of the messages that you want to delete.
 - 3) Click **Delete**.
 - b. Optional: To export all the remaining messages in a failed sequence, complete the following steps:
 - 1) In the main pane of the Outbound sequence collection form, select the check box next to the name of the failed sequence.

- 2) Click **Export unsent messages**. All remaining messages in the sequence are exported to a compressed file.
4. Close or terminate the failed sequence.

Note: In the WS-ReliableMessaging Version 1.1 specification, a sequence can be closed rather than terminated. This allows the final ACK state to be sent from the reliable messaging destination to the reliable messaging source. In the WS-ReliableMessaging Version 1.0 specification this does not happen, so the final ACK state might not be known at the reliable messaging source. For more information about the distinction between close and terminate, see Outbound sequence collection.

- a. In the main pane of the Outbound sequence collection form, select the check box next to the name of the failed sequence.
 - b. Click **Close sequence** or **Terminate sequence**.
5. Delete the failed sequence.
 - a. In the main pane of the Outbound sequence collection form, select the check box next to the name of the failed sequence.
 - b. Click **Delete sequence**.

WS-ReliableMessaging troubleshooting tips

Tips for troubleshooting your WS-ReliableMessaging configuration.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log , SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

To help you identify and resolve problems with WS-ReliableMessaging, you can use the TCP/IP monitor to view the messages that are flowing between your client applications and reliable messaging enabled Web services. You can also use the WebSphere Application Server trace and logging facilities as described in Setting up component trace (CTRACE).

To enable trace for WS-ReliableMessaging, set the application server trace string as follows:

- For either of the managed qualities of service:

```
org.apache.sandesha2*=all=enabled:com.ibm.ws.websvcs.rm*=all=enabled:org.apache.axis2*=all=enabled:com.ibm.ws.sib.wsrn*=all=enabled
```

- For the Unmanaged Non-Persistent quality of service:

```
org.apache.sandesha2*=all=enabled:com.ibm.ws.websvcs.rm*=all=enabled:org.apache.axis2*=all=enabled
```

If you encounter a problem that you think might be related to WS-ReliableMessaging, 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-ReliableMessaging is provided in “WS-ReliableMessaging known restrictions” on page 273.

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-ReliableMessaging component is CWSKA.

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:

- “When you examine the runtime state of inbound or outbound sequences you might see more sequences than you expect, due to sequence reallocation”
- “When you examine the runtime state of inbound or outbound sequences you might see multiple entries for each sequence”
- “You get runtime errors when you migrate persisted WS-ReliableMessaging messages from Version 6.1.0.9 or 6.1.0.11 of the Feature Pack for Web Services to a later version of the product”
- “When an application server starts, a messaging engine used for reliable messaging is reported as unavailable” on page 270
- “A client application is unable to invoke a reliable messaging enabled web service” on page 270
- “A sequence is not established, and therefore WS-ReliableMessaging cannot ensure messages are transmitted” on page 270
- “A sequence is established but cannot be used, and therefore WS-ReliableMessaging cannot ensure messages are transmitted” on page 271
- “The reliable messaging managed store is not initialized because the policy set binding is not complete or not valid” on page 271
- “Reliable messaging is interrupted because a server is unavailable” on page 272
- “Socket timeout errors are received when running multiple reliable messaging client applications in a cluster” on page 272
- “A message is not recovered after a server becomes unavailable, even with the managed persistent quality of service” on page 272
- “When using reliable messaging with a persistent WS-I RSP profile and WS-SecureConversation, an exception message states that the security context token is not valid” on page 272
- “A servant region experiences a timeout abend when you are using a managed quality of service” on page 272

When you examine the runtime state of inbound or outbound sequences you might see more sequences than you expect, due to sequence reallocation

If a sequence is reallocated, the original and new sequences are both visible. Ignore the multiple entries.

When you examine the runtime state of inbound or outbound sequences you might see multiple entries for each sequence

This is because, although reliable messaging only binds to one messaging engine, the runtime panel is calculating and displaying the sequence information once for every servant region. Ignore the duplicate entries. Note that the slight differences in the statistics being displayed for each duplicate entry is due to the entries being created sequentially, while polling for messages continues.

You get runtime errors when you migrate persisted WS-ReliableMessaging messages from Version 6.1.0.9 or 6.1.0.11 of the Feature Pack for Web Services to a later version of the product

If you are migrating from WebSphere Application Server Version 6.1, and you are using Version 6.1.0.9 or 6.1.0.11 of the Feature Pack for Web Services, and your configuration includes WS-ReliableMessaging configured for the managed persistent quality of service, you need to remove all persisted messages before you migrate.

Each message is persisted as part of a sequence that is currently being processed. To remove all persisted messages, use the administrative console to complete the following steps:

1. Navigate to the Inbound sequence collection runtime panel for your reliable messaging application.

2. Select all the inbound sequences, then click **delete sequence and messages** to delete the sequences.
3. Navigate to the Outbound sequence collection runtime panel, then repeat the previous steps for the outbound sequences.

When an application server starts, a messaging engine used for reliable messaging is reported as unavailable

When you use reliable messaging with a managed quality of service, you might see the following exception message when your application server starts:

```
CWSIT0019E: No suitable messaging engine is available on bus yourBus that matched the specified connection properties
```

In a network deployment environment, this can occur because the messaging engine is on an application server or cluster member that has started later than the server that hosts your reliable messaging application. In this case you need do nothing but wait; reliable messaging will keep trying to connect until the messaging engine becomes available.

If you suspect there is an underlying problem, for example the bindings are incorrect or the server that hosts the messaging engine is not going to start, complete the following checks:

- Check that the specified messaging engine and service integration bus exist.
- Check the system out log to ensure that the server that hosts the messaging engine has started.

A client application is unable to invoke a reliable messaging enabled web service

If your client application is unable to invoke a reliable messaging enabled web service, you can use the TCP-IP monitor to view the messages that are flowing between the client and the service. You should also check the following:

- The endpoint is available.
- The service is running.
- The service has been invoked.
- WS-ReliableMessaging is running.
- WS-ReliableMessaging is correctly configured. In particular, for either of the managed qualities of service, check that you have configured a valid binding to a service integration bus and messaging engine, and that the messaging engine is running. For more information, see *Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the administrative console* or *Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the wsadmin tool*.
- There are not too many applications sharing a single messaging engine.

Note: When many applications use the same messaging engine, it can impact performance. Factors to consider include the number of applications that are already binding to the messaging engine, the CPU utilization, and the message throughput. To improve performance for a single server configuration, create a new messaging engine to bind to your application.

A sequence is not established, and therefore WS-ReliableMessaging cannot ensure messages are transmitted

After a sequence has been established, WS-ReliableMessaging provides retransmission of messages to a service. However if the sequence is not established then the messages are not transmitted to the service and a message similar to the following example is displayed:

```
org.apache.axis2.AxisFault: The Create Sequence request has been refused by the RM Destination
```


The initial createSequence message has been refused. This is propagated back, and causes the client to fail. For information about CreateSequence and CreateSequenceRefused, see the WS-ReliableMessaging: supported specifications and standards.

You might also see a subsequent message to help explain why the request has been refused. For example:

```
Caused by: javax.xml.ws.soap.SOAPFaultException: com.ibm.ws.sib.wsruntime.exceptions.WSRMRuntimeException:
CWSJZ0202I: A messaging engine connection is unavailable for bus myBus.
```

There is a problem with your reliable messaging configuration. Complete the following checks:

- Check that the policy sets are correctly applied. Specifically, check that the destination has reliable messaging correctly enabled.
- Check the logs for server-side problems.
- For the managed persistent quality of service, check that the associated messaging engine is available.

For further checks that might help you resolve the problem, see also the following troubleshooting tips:

- “A sequence is established but cannot be used, and therefore WS-ReliableMessaging cannot ensure messages are transmitted.”
- “A client application is unable to invoke a reliable messaging enabled web service” on page 270.

A sequence is established but cannot be used, and therefore WS-ReliableMessaging cannot ensure messages are transmitted

If you get an exception such as the following exception, then the sequence is established but cannot be used:

```
javax.xml.ws.WebServiceException: org.apache.axis2.AxisFault: The value of wsr:Identifier is not a known Sequence identifier.
```

The most common reason is that you are working in a clustered environment but your server-side policy set specifies the unmanaged non-persistent quality of service. For example: The WS-I RSP default policy set specifies the unmanaged non-persistent quality of service. To use reliable asynchronous messaging in a clustered environment, you must use a managed quality of service to enable the cluster members to correlate reliable messaging state. To do this, either use the WS-I RSP ND default policy set, or modify your custom policy set so that the WS-ReliableMessaging policy specifies a managed quality of service, and an associated binding to a service integration bus and messaging engine. For information about how to do this, see *Configuring a WS-ReliableMessaging policy set by using the administrative console* and *Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the administrative console*.

For further checks that might help you resolve the problem, see also the following troubleshooting tips:

- “A sequence is not established, and therefore WS-ReliableMessaging cannot ensure messages are transmitted” on page 270.
- “A client application is unable to invoke a reliable messaging enabled web service” on page 270.

The reliable messaging managed store is not initialized because the policy set binding is not complete or not valid

If your policy set specifies a managed quality of service, but you have not specified a binding to a messaging engine to support that quality of service, you get the following exception message:

```
CWSKA0102E: The managed web services reliable messaging storage manager could not be initialized because the policy set binding
was incomplete or invalid.
```

Perhaps you have attached a managed policy set to your application, and used the default bindings (which do not support the managed qualities of service). You must create a new binding for your application that specifies a service integration bus and messaging engine to support the managed qualities of service. To

do this, see [Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the administrative console](#).

Reliable messaging is interrupted because a server is unavailable

Clustering offers maximum protection against servers becoming unavailable. It provides highly available service endpoints, and (through the service integration bus) high availability of the reliable messaging layer.

For more information about configuring high availability for web services and messaging engines, see [Balancing workloads and ..ae/tjj0042_.dita](#).

Socket timeout errors are received when running multiple reliable messaging client applications in a cluster

When many applications use the same messaging engine, it can impact performance and in some cases lead to timeout errors.

For more information about configuring high availability for web services and messaging engines, see [Balancing workloads and ..ae/tjj0042_.dita](#).

A message is not recovered after a server becomes unavailable, even with the managed persistent quality of service

When the reliable messaging layer receives a request message, it sends an acknowledgement then delivers the message to the target service. There is a marginal possibility that the server hosting the reliable messaging layer might become unavailable after the request message has been acknowledged and before it has been delivered. In this case, the message is only recovered if you are using in-order delivery as well as managed persistent quality of service. To specify in-order delivery, select the WS-ReliableMessaging policy option to “Deliver messages in the order that they were sent” as described in [Configuring the WS-ReliableMessaging policy](#).

Note: There is a performance overhead in using in-order delivery, because messages are held in a queue until they can be delivered in order. However, where the highest level of reliability is required, you should always specify in-order delivery in conjunction with the managed persistent quality of service.

When using reliable messaging with a persistent WS-I RSP profile and WS-SecureConversation, an exception message states that the security context token is not valid

When you use a persistent WS-I RSP policy set, which includes WS-SecureConversation, if the scoping security context token is expired when the server is restarted then WS-ReliableMessaging cannot resend its messages and system messages are written to the log file stating that the reliable messaging sequence was not secured using the correct security token. For example:

```
CWWS7215E: Cannot get valid security context token from the cache.
```

To ensure that the scoping security context token does not expire before WS-ReliableMessaging can recover and resend its messages, complete the following task: [Configuring WS-SecureConversation to work with WS-ReliableMessaging](#).

A servant region experiences a timeout abend when you are using a managed quality of service

A timeout abend could be caused by the value of the sib.wsrn.tokenLockTimeout custom property being set too high. This property is set on the messaging engine that is specified on the WS-ReliableMessaging

policy binding. The value should be less than the amount of time that the control region waits before ending an inactive servant region. Refer to the `../ae/rjj_custom_props.dita` topic for more information about this property.

WS-ReliableMessaging known restrictions

The main known restrictions that apply to the implementation of WS-ReliableMessaging in WebSphere Application Server.

- “IBM HTTP Server is not supported as a proxy server with WS-ReliableMessaging”
- “A WS-ReliableMessaging policy can only be applied at application or service level.”
- “WS-SecureConversation must expect the WS-ReliableMessaging headers to be signed”

IBM HTTP Server is not supported as a proxy server with WS-ReliableMessaging

When WS-ReliableMessaging is enabled, you cannot use IBM HTTP Server as a proxy server to route responses back to an individual server in a clustered environment. If you use IBM HTTP Server in this scenario, replies to protocol messages, such as WS-ReliableMessaging CreateSequenceResponse messages, are not routed back to the originating server. This is because WS-ReliableMessaging protocol messages use WS-Addressing affinity headers to ensure that the protocol responses are routed back to the originating server. These affinity headers are supported in IBM WebSphere Application Server proxy server, but are not supported in IBM HTTP Server.

Use the IBM WebSphere Application Server proxy server to route messages from and back to the cluster, instead of IBM HTTP Server.

A WS-ReliableMessaging policy can only be applied at application or service level.

You can only apply a WS-ReliableMessaging policy at application level or service level.

If you apply reliable messaging at service level, then all services must use the same WS-ReliableMessaging policy and bindings values.

You can attach any policy set at operation level. For a policy set that includes the WS-ReliableMessaging policy, attachment at the operation level configures the other components of the policy set (for example WS-Security and WS-Addressing) but any WS-ReliableMessaging configuration at operation level is ignored.

WS-SecureConversation must expect the WS-ReliableMessaging headers to be signed

Although secure conversation allows message headers to remain unsigned, the reliable messaging policy requires the reliable messaging headers to be signed. If you want to use secure conversation and reliable messaging policies in the same policy set, the secure conversation bindings must be configured to require that the reliable messaging headers are signed.

The reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND) are specifically designed and configured to use secure conversation and reliable messaging in the same policy set. If you use a copy of one of the reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND), no further configuration of the secure conversation bindings is required. Otherwise, see [Configuring WS-SecureConversation to work with WS-ReliableMessaging](#).

Troubleshooting WSIF

Use this overview task to help resolve a problem that you think is related to the Web Services Invocation Framework (WSIF).

Before you begin

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

For information about resolving WebSphere Application Server-level problems, see [Diagnosing problems \(using diagnosis tools\)](#).

About this task

To identify and resolve WSIF-related problems, you can use the standard WebSphere Application Server trace and logging facilities. If you encounter a problem that you think might be related to WSIF, you can check for error messages in the WebSphere Application Server administrative console, and in the application server `stdout.log` file. You can also enable the application server debug trace to provide a detailed exception dump.

The documentation also includes details of common WSIF-related problems, specific WSIF troubleshooting tips and known WSIF restrictions.

Procedure

1. Trace and log WSIF.

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. For more information, see [“Tracing and logging WSIF.”](#)

2. Check for error messages about WSIF.

A list of the WSIF runtime system messages, with details of what each message means, is provided in [“WSIF \(Web Services Invocation Framework\) messages”](#) on page 275.

Check in the `SYSPRINT` or `SYSOUT` log.

3. Check the list of major WSIF activities, with advice on common problems associated with each activity

For more information, see [“Web Services Invocation Framework troubleshooting tips”](#) on page 276.

4. Check the set of specific tips to help you troubleshoot problems you experience with WSIF.

For more information, see [“Web Services Invocation Framework troubleshooting tips”](#) on page 276.

5. Check the list of the main known restrictions that apply when using WSIF.

For more information, see [“WSIF - Known restrictions”](#) on page 280.

Tracing and logging 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.

Before you begin

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using

HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

About this task

To enable trace and logging for WSIF, complete either or both of the following steps:

Procedure

- 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.

To trace the WSIF API, specify the following trace string:

```
wsif=all=enabled
```

For more information, see Setting up component trace (CTRACE).

- Enable the WSIF SimpleLog utility, through which you can run trace when you are using WSIF outside of WebSphere Application Server.

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.

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

WSIF (Web Services Invocation Framework) messages

WebSphere 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. Here is a list of the WSIF runtime system messages, with details of what each message means.

WSIF0001E: An extension registry was not found for the element type “{0}”

Explanation: Parameters: {0} element type. No extension registry was found for the element type specified.

Action: Add the appropriate extension registry to the port factory in your code.

WSIF0002E: A failure occurred in loading WSDL from “{0}”

Explanation: Parameters: {0} location of the WSDL file. The WSDL file could not be found at the location specified or did not parse correctly

Action: Check that the location of the WSDL file is correct. Check that any network connections required are available. Check that the WSDL file contains valid WSDL.

WSIF0003W: An error occurred finding pluggable providers: {0}

Explanation: Parameters: {0} specific details about the error. There was a problem locating a WSIF pluggable provider using the J2SE 1.3 JAR file extensions to support service providers architecture. The WSIF trace file will contain the full exception details.

Action: Verify that a `META-INF/services/org.apache.wsif.spi.WSIFProvider` file exists in a provider jar, that each class referenced in the `META-INF` file exists in the class path, and that each class implements `org.apache.wsif.spi.WSIFProvider`. The class in error will be ignored and WSIF will continue locating other pluggable providers.

WSIF0004E: WSDL contains an operation type “{0}” that is not supported for “{1}”

Explanation: Parameters: {0} name of the operation type specified. {1} name of the portType for the operation. An operation type that is not supported has been specified in the WSDL.

Action: Remove any operations of the unsupported type from the WSDL. If the operation is required then make sure all messages have been correctly specified for the operation.

WSIF0005E: An error occurred when invoking the method “{1}” . (“{0}”)

Explanation: Parameters: {0} name of communication type. For example EJB or Apache SOAP. {1} name of the method that failed. An error was encountered when invoking a method on the web service using the communication shown in brackets.

Action: Check that the method exists on the web service and that the correct parts have been added to the operation as described in the WSDL. Network problems might be a cause if the method is remote and so check any required connections.

WSIF0006W: Multiple WSIFProvider found supporting the same namespace URI “{0}” . Found (“{1}”)

Explanation: Parameters: {0} the namespace URI. {1} a list of the WSIFProvider found.. There are multiple org.apache.wsif.spi.WSIFProvider classes in the service provider path that support the same namespace URI.

Action: A following WSIF00071 message will be issued notifying which WSIFProvider will be used. Which WSIFProvider is chosen is based on settings in the wsif.properties file, or if not defined in the properties, the last WSIFProvider found will be used. See the wsif.properties file for more details on how to define which provider should be used to support a namespace URI.

WSIF00071: Using WSIFProvider “{0}” for namespaceURI “{1}”

Explanation: Parameters: {0} the classname of the WSIFProvider being used. {1} the namespaceURI the provider will be used to support.. Either a previous WSIF0006W message has been issued or the SetDynamicWSIFProvider method has been used to override the provider used to support a namespaceURI.

Action: None. See also WSIF0006W.

WSIF0008W: WSIFDefaultCorrelationService removing correlator due to timeout. ID:“{0}”

Explanation: Parameters: {0} the ID of the correlator being removed from the correlation service. A stored correlator is being removed from the correlation service due to its timeout expiring.

Action: Determine why no response has been received for the asynchronous request within the timeout period. The wsif.asyncrequest.timeout property of the wsif.properties file defines the length of the timeout period.

WSIF0009I: Using correlation service - “{0}”

Explanation: Parameters: {0} the name of the correlation service being used. This identifies the name of the correlation service that will be used to process asynchronous requests.

Action: None. If a correlation service other than the default WSIF supplied one is required, ensure that it is correctly registered in the JNDI java:comp/wsif/WSIFCorrelationService namespace.

WSIF0010E: Exception thrown while processing asynchronous response - “{0}”

Explanation: Parameters: {0} the error message string of the exception. While processing the response from an executeRequestResponseAsync call an exception was thrown.

Action: Use the exception error message string to determine the cause of the error. The WSIF trace will have more details on the error including the exception stack trace.

WSIF0011I: Preferred port “{0}” was not available

Explanation: Parameters: {0} the user's preferred port. The preferred port set by the user on org.apache.wsif.WSIFService is not available

Action: None unless this message appears for long periods of time in which case the user might want to pick a different port as their preferred port.

Web Services Invocation Framework troubleshooting tips

A set of specific tips to help you troubleshoot problems you experience with the Web Services Invocation Framework (WSIF).

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log ,

SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

For information about resolving WebSphere Application Server-level problems, see *Diagnosing problems (using diagnosis tools)*.

To identify and resolve WSIF-related problems, you can use the standard WebSphere Application Server trace and logging facilities. If you encounter a problem that you think might be related to WSIF, you can check for error messages in the WebSphere Application Server administrative console, and in the application server `stdout.log` file. You can also enable the application server debug trace to provide a detailed exception dump.

A list of the WSIF runtime system messages, with details of what each message means, is provided in “WSIF (Web Services Invocation Framework) messages” on page 275.

A list of the main known restrictions that apply when using WSIF is provided in “WSIF - Known restrictions” on page 280.

Here is a checklist of major WSIF activities, with advice on common problems associated with each activity:

Create service

A handcrafted Web Services Description Language (WSDL) file can cause numerous problems. To help ensure that your WSDL file is valid, use a tool such as WebSphere Studio Application Developer (WSAD) to create your web service. For more details about creating services with WSDL in a distributed or a z/OS environment, read the WSIF and WSDL chapter of *Developing and deploying applications* PDF book.

Define transport mechanism

For the Java Message Service (JMS), check that you have set up the Java Naming and Directory Interface (JNDI) correctly, and created the necessary connection factories and queues.

For SOAP, make sure that the deployment descriptor file `dds.xml` is correct - preferably by creating it using WebSphere Studio Application Developer (WSAD) or similar tooling.

Create client - Java code

Follow the correct format for creating a WSIF service, port, operation and message. For examples of correct code, refer to the Address Book sample in the *Developing and deploying applications* PDF book.

Compile code (client and service)

Check that the build path against code is correct, and that it contains the correct levels of JAR files.

Create a valid EAR file for your service in preparation for deployment to a web server.

Deploy service

When you install and deploy the service EAR file, check carefully any messages given when the service is deployed.

Server setup and start

Make sure that the WebSphere Application Server `server.policy` file (in the `/properties` directory) has the correct security settings. For more information about setting up security, see the Enabling security for WSIF topic in the *Securing applications and their environment* PDF book.

WSIF setup

Check that the `wsif.properties` file is correctly set up. For more information about this file, refer to the Maintaining the WSIF properties file topic in the *Administering applications and their environment* PDF book.

Run client

Either check that you have defined the class path correctly to include references to your client classes, WSIF JAR files and any other necessary JAR files, or (preferably) run your client by using the WebSphere Application Server launchClient tool.

Either check that you have defined the class path correctly to include references to your client classes, WSIF JAR files and any other necessary JAR files, or (preferably) run your client by using the WebSphere Application Server launch client tool. For more information about this tool, refer to the Running application clients chapter of the *Developing and deploying applications* PDF book.

Here is a set of tips to help you troubleshoot commonly-experienced problems:

- “No class definition errors received when running client code”
- “Cannot find WSDL error”
- “Your web service EAR file does not install correctly onto the application server”
- “There is a permissions problem or security error” on page 279
- “Using WSIF with multiple clients causes a SOAP parsing error” on page 279
- “JNDI lookup errors occur when using the same names for JMS messaging queues and queue connection factories that run on application servers on different machines” on page 279
- “A JAX-RPC client running on WebSphere Application Server Version 5 uses SOAP over JMS to invoke a web service running on a Version 5 application server. No user ID or password is required on the target MQ Series queue. After the application server is migrated to Version 6, and using Version 6 default messaging, client requests fail because basic authentication is now enabled” on page 279
- “The current WSIF default SOAP provider (the IBM Web Service SOAP provider) does not interoperate fully with services that are running on the former (Apache SOAP) provider” on page 280

“No class definition” errors received when running client code

This problem usually indicates an error in the class path setup. Check that the relevant JAR files are included.

“Cannot find WSDL” error

Some likely causes are:

- The application server is not running.
- The server location and port number in the WSDL are not correct.
- The WSDL is badly formed (check the error messages in the application server `stdout.log` file).
- The application server has not been restarted since the service was installed.

You might also try the following checks:

- Can you load the WSDL into your web browser from the location specified in the error message?
- Can you load the corresponding WSDL binding files into your Web browser?

Your web service EAR file does not install correctly onto the application server

The EAR file is probably badly formed. Verify the installation by completing the following steps:

- For an EJB binding, run the WebSphere Application Server tool `\bin\dumpnamespace`. This tool lists the current contents of the JNDI directory.
- For a SOAP over HTTP binding, open the `http://pathToServer/WebServiceName/admin/list.jsp` page (if you have the SOAP administration pages installed). This page lists all currently installed web services.
- For a SOAP over JMS binding, complete the following checks:
 - Check that the queue manager is running.
 - Check that the necessary queues are defined.
 - Check the JNDI setup.
 - Use the “display context” option in the `jmsadmin` tool to list the current JNDI definitions.
 - Check that the Remote Procedure Call (RPC) router is running.

There is a permissions problem or security error

Check that the WebSphere Application Server `server.policy` file (in the `/properties` directory) has the correct security settings. For more information about setting up security, see the Enabling security for WSIF topic in the *Securing applications and their environment* PDF book.

Using WSIF with multiple clients causes a SOAP parsing error

Before you deploy a web service to WebSphere Application Server, you must decide on the scope of the web service. The deployment descriptor file `dds.xml` for the web service includes the following line:

```
<isd:provider type="java" scope="Application" .....
```

You can set the `Scope` attribute to `Application` or `Session`. The default setting is `Application`, and this value is correct if each request to the web service does not require objects to be maintained for longer than a single instance. If `Scope` is set to `Application` the objects are not available to another request during the execution of the single instance, and they are released on completion. If your web service needs objects to be maintained for multiple requests, and to be unique within each request, you must set the scope to `Session`. If `Scope` is set to `Session`, the objects are not available to another request during the life of the session, and they are released on completion of the session. If scope is set to `Application` instead of `Session`, you might get the following SOAP error:

```
SOAPException: SOAP-ENV:ClientParsing error, response was:  
FWK005 parse might not be called while parsing.;  
nested exception is:
```

```
[SOAPException: faultCode=SOAP-ENV:Client; msg=Parsing error, response was:
```

```
FWK005 parse might not be called while parsing.;  
targetException=org.xml.sax.SAXException:  
FWK005 parse might not be called while parsing.]
```

JNDI lookup errors occur when using the same names for JMS messaging queues and queue connection factories that run on application servers on different machines

You should not use the same names for messaging queues and queue connection factories that run on application servers on different machines, because WSIF always checks first for JMS destinations locally, and only uses the full JNDI reference if it cannot find the destination locally. For example, if you run a web service on a remote machine, and have an application server running locally that uses the same names for the messaging queues and queue connection factories, then WSIF will find and use the local queues even if the remote JNDI destination is provided in full in the WSDL service definition.

A JAX-RPC client running on WebSphere Application Server Version 5 uses SOAP over JMS to invoke a web service running on a Version 5 application server. No user ID or password is required on the target MQ Series queue. After the application server is migrated to Version 6, and using Version 6 default messaging, client requests fail because basic authentication is now enabled

The problem appears as a log message:

```
SibMessage W [:] CWSIT0009W: A client request failed in the application server  
with endpoint <endpoint_name> in bus <your_bus> with reason: CWSIT0016E:  
The user ID null failed authentication in bus <your_bus>.
```

For the steps to take to resolve the problem, see the following service integration technologies troubleshooting tip: “After you migrate an application server to WebSphere Application Server Version 6, existing web services clients can no longer use SOAP over JMS to access services hosted on the migrated server.” This tip can be found in the Tips for troubleshooting service integration bus security section.

The current WSIF default SOAP provider (the IBM Web Service SOAP provider) does not interoperate fully with services that are running on the former (Apache SOAP) provider

This is because the IBM Web Service SOAP provider is designed to interoperate fully with a JAX-RPC compliant web service, and Apache SOAP cannot provide such a service. To enable interoperation, modify either your web service or the WSIF default SOAP provider. For information about how to modify your provider, read the chapter “WSIF SOAP provider: working with existing applications” in the *Developing and deploying applications PDF* book.

WSIF - Known restrictions

The known restrictions that apply when using WSIF include restrictions on threading, on SOAP headers and unreferenced attachments, and on data type mappings.

Threading

WSIF is not thread-safe.

External Standards

WSIF supports:

- SOAP Version 1.1 (not 1.2 or later).
- WSDL Version 1.1 (not 1.2 or later).

WSIF does not provide WS-I compliance, and it does not support the Java API for XML-based Remote Procedure Calls (JAX-RPC) Version 1.1 (or later).

Full schema parsing

WSIF does not support full schema parsing. For example, Web Services Description Language (WSDL) references in complex types in the schema are not handled, and attributes are not handled.

XML Schema “redefine” elements are not handled and are ignored.

SOAP

WSIF does not support:

- SOAP headers that are passed as <parts>.
- Unreferenced attachments in SOAP responses, or the scenarios detailed in SOAP attachments - scenarios that are not supported.
- Document Encoded style SOAP messages.

Note: This is not primarily a WSIF restriction. Although you can specify Document Encoded style in WSDL, it is not generally considered to be a valid option and is not supported by the Web Services Interoperability Organization (WS-I).

SOAP provider interoperability

The current WSIF default SOAP provider (the IBM Web Service SOAP provider) does not fully interoperate with services that are running on the former (Apache SOAP) provider. This is because the IBM Web Service SOAP provider is designed to interoperate fully with a JAX-RPC compliant web service, and Apache SOAP cannot provide such a service. For information on how to overcome this restriction, see WSIF SOAP provider: working with existing applications.

WSIF support for SOAP faults is restricted to SOAP faults originating from a web service that uses the IBM Web Service SOAP provider.

Note: This is not primarily a WSIF restriction. The current SOAP faults specification does not prescribe how to encode a SOAP fault so that it maps to a Java exception. Consequently, each web service runtime environment currently chooses its own SOAP fault format. The IBM Web Service SOAP provider can understand its own response SOAP faults, but not the SOAP faults from another provider.

Data type mappings

The current WSIF default SOAP provider (the IBM Web Service SOAP provider) conforms to the JAX-RPC type mapping rules that were finalized after the former (Apache SOAP) provider was created. The majority of data types are mapped the same way by both providers. The exceptions are: `xsd:date`, `xsd:dateTime`, `xsd:hexBinary` and `xsd:QName`. Both client and service must use the same mapping rules if any of these four data types are used. Below is a table detailing the mapping rules for these four data types:

Table 15. Mapping rules for the four data types that are mapped differently by Apache SOAP and JAX-RPC.

Column 1 specifies the XML data type, column 2 specifies the equivalent data type for Apache SOAP, and column 3 specifies the equivalent data type for JAX-RPC.

XML Data Type	Apache SOAP Java Mapping	JAX-RPC Java Mapping
<code>xsd:date</code>	<code>java.util.Date</code>	Not supported
<code>xsd:dateTime</code>	Not supported	<code>java.util.Calendar</code>
<code>xsd:hexBinary</code>	Hexadecimal string	<code>byte []</code>
<code>xsd:QName</code>	<code>org.apache.soap.util.xml.QName</code>	<code>javax.xml.namespace.QName</code>

Arrays and complex types

WSIF does not support general complex types, it only handles complex types that map to Java Beans. To use schema complex types, you must write your own custom serializers. The specific complex type and array support for WSIF outbound invocation of web services is as follows:

- WSIF supports Java classes generated by WebSphere Studio Application Developer - Integration Edition (WSAD-IE) message generators (the usual case when WSDL files are downloaded from somewhere else). The WSAD-IE-based generation happens automatically when you use the BPEL editor, or the generation actions available on the Enterprise Services context menu, or the Business Integration toolbar.
- WSIF does not support Java beans generated by other tools, including the base WSAD tool.
- For WSAD-IE generated Java beans, attributes defined in the WSDL do not work. That is to say that these attributes, although they appear in the Java beans generated to represent the complex type, do not appear in the SOAP request created by WSIF.
- WSIF does not support arrays when they are a field of a Java bean. That is to say, WSIF only supports an array that is passed in as a named `<part>`. If an array is wrapped inside a Java bean, the array is not serialized in the same way.

Object Serialization

WSIF does not support serialization of objects across different releases.

Asynchronous invocation

WSIF supports synchronous invocation for all providers. For the JMS and the SOAP over JMS providers, WSIF also supports asynchronous invocation. You should call the `supportsAsync()` method before trying to execute an asynchronous operation.

The EJB provider

The target service of the WSIF EJB provider must be a remote-home interface, it cannot be an EJB local-home interface. In addition, the EJB stub classes must be available on the client class path.

Running outside WebSphere Application Server

WSIF is not supported for use outside WebSphere Application Server.

UDDI registry troubleshooting

Common errors that can occur when you use the UDDI registry include database and data source errors, JavaScript syntax errors on pages in the UDDI user console, a UDDI node not appearing in the administrative console list of available nodes, and not being able to issue UDDI requests.

About this task

During UDDI registry use, messages to report events or errors might be issued. Use these messages first to resolve any problems. For further assistance, review the following list of troubleshooting tips.

For more details about a problem, enable trace for UDDI. You can enable or disable trace by using the administrative console. The component for the UDDI registry is `com.ibm.uddi`. You can enable trace for the UDDI registry at several levels of granularity. For example, to enable all UDDI registry tracing, specify the following:

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

The following list contains some problems that might occur when you set up or use the UDDI registry, and suggested solutions.

Procedure

- The first startup of the UDDI application might take time to complete.
 - When you start the UDDI registry application for the first time with a new UDDI registry database, it must complete 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 predefined data and entities, and can take time to complete. This is expected behavior, and affects only the first startup of the UDDI application.
 - If you use the `wsadmin` utility to issue a command to start the UDDI application, depending on your TCP timeout settings, this request might time out while it waits for UDDI to complete initialization. UDDI initialization and starting the UDDI application continue to complete normally; they are not affected by this timeout.
- JavaScript syntax errors might occur on several web pages in the UDDI user console. This problem is caused by a limitation about URL rewriting. To avoid this limitation, 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.
- For a remote DB2 database, if attaching to the remote system causes problems, one possible cause might be IP addressing. This situation does not occur when the remote system uses a static IP address. However, if the remote system uses Dynamic Host Configuration Protocol (DHCP), each system must be aware of the subnet mask of the other system.
- The UDDI node is not in the list of available nodes on the administrative console. Check that the UDDI application is started on the relevant node and server.
- You cannot issue UDDI requests, for example, you start the UDDI user console but errors occur when you try to publish or inquire. Consider the following reasons:
 - The database is not currently loaded or configured. Check the output from creating the database.
 - The database is not correctly configured. Check that the JDBC provider and data source definitions are correct by clicking **Test Connection** on the administrative console.
 - The UDDI node is not initialized. Check the UDDI node page on the administrative console. If the status of the node is not activated or deactivated, set any policies or properties, then try to initialize the node.
 - The UDDI node is currently deactivated while UDDI runtime settings are updated. Check the UDDI node page on the administrative console. If the status of the node is deactivated, wait for the status to become activated, then try the request again.
 - There are problems with your DB2 setup. In particular, check that the DB2 bind utility has run and that a TEMP database is defined in the DB2 location. See the topic about using the DB2 Universal JDBC Driver to access DB2 for z/OS.
- Ensure that when you define the data source, you specify the location name for the database name, rather than the database name that you supplied to the `createddl.sh` script (see the topic about creating a DB2 for z/OS database for the UDDI registry). For example, specify `L0C1`, not `UDDI30`.

- When you try to save a business entity in a UDDI registry that runs on WebSphere Application Server Version 7.0 or later, but the registry uses a Cloudscape or Apache Derby database that was created using an earlier version of the product, the following error might occur:

Serious technical error has occurred while processing the request.

This error occurs if a UDDI database currently uses Apache Derby Version 10.2 or later and you have upgraded the servers to WebSphere Application Server Version 7.0 or later. You must migrate the UDDI database. See the topic about migrating a UDDI database that uses Apache Derby.

What to do next

If you have not resolved your problem, see the problem determination information on the WebSphere Application Server support web page.

Bus-enabled web services troubleshooting tips

Use this set of specific tips to help you troubleshoot problems you experience with service integration bus-enabled web services.

To help you identify and resolve bus-enabled web services problems, use the WebSphere Application Server trace and logging facilities as described in [Setting up component trace \(CTRACE\)](#).

To enable trace for bus-enabled web services, set the application server trace string to `com.ibm.ws.sib.webservices.*=all=enabled`. If you encounter a problem that you think might be related to bus-enabled web services, 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 bus-enabled web services is provided in [Bus-enabled web services: Known restrictions](#).

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 bus-enabled web services component is `CWSWS`.

The topic [Messages](#) 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.

Security tips:

- “Your bus-enabled web services are unable to connect to a secure service integration bus” on page 284
- “A JAX-RPC client running on WebSphere Application Server Version 5.1 uses SOAP over JMS to invoke a web service running on a Version 5 application server. No user ID or password is required on the target MQ Series queue. After the application server is migrated to a later version, and to use default messaging, client requests fail because basic authentication is now enabled” on page 286
- “If the bus needs to pass messages through an authenticating proxy server to retrieve WSDL documents, then you must use command-line tools to retrieve the WSDL” on page 287
- “You are trying to send a SOAP over HTTPS message, and you are receiving a Malformed URLErrorException error” on page 288
- “You are password-protecting a web service operation, but when you install the `sibwsauthbean.ear` file, an error message is displayed in the WebSphere Application Server administrative console detailing a Java Naming and Directory Interface (JNDI) problem” on page 288

Non-security tips:

- “The service integration bus times out while an outbound service is waiting for the response from a target service” on page 285

- “You must manually install one of the bus-enabled web services applications or resources” on page 285
- “You have a client application that works under WebSphere Application Server Version 5.1, but in later versions you get problems caused by poorly-formed requests or responses” on page 285
- “You unsuccessfully attempt to create an Informix database for use with the SDO repository, and receive the message No Transaction Isolation on non-logging databases.” on page 286
- “You have an inbound service that, according to the administrative console, is published to a UDDI registry. When you check the UDDI registry you discover that the service is not listed there. When you attempt to use the administrative console to republish the service to UDDI, the attempt is unsuccessful” on page 286
- “If you are using JMS to connect to a remote bus, extra configuration is required to allow web service clients to connect to the bus” on page 287
- “You pass a large attachment through the service integration bus and you get an out-of-memory error in the Java virtual machine” on page 287
- “When you try to create a new endpoint listener configuration by using the administrative console, and click New on the endpoint listener collection panel, the Please wait icon is displayed but the target panel does not appear” on page 288
- “When you try to create a new inbound service through the administrative console, the drop-down list of destinations is empty and the wizard stops you at step 1 with the error message A destination must be selected” on page 288
- “You experience problems with handling SOAP messages with attachments” on page 288
- “You get JNDI lookup errors when you use the same names for JMS messaging queues and queue connection factories that run on application servers on different machines” on page 288
- “You are getting SOAP fault messages, but cannot determine the precise problem from the fault message” on page 288
- “You get listener port timeout errors when large messages are passed by using the SOAP over JMS endpoint listener” on page 289

Your bus-enabled web services are unable to connect to a secure service integration bus

When the bus-enabled web services component cannot connect to a secure bus, the following error message is issued during the server start (in the application server `SystemOut.log` file) when the service integration resource adapter attempts to connect to the bus destination:

```
CWSIV0801E: The exception javax.resource.ResourceException:
CWSIV0958E: The authorization exception
com.ibm.wsspi.sib.core.exception.SINotAuthorizedException:
CWSIP0302E: A user HostServer is not authorized to access the messaging engine
xyzNode01.server1-xyz on bus xyz. was thrown while attempting to create a
connection to messaging engine 221C86B845BE5E8B using the activation
specification [<activation_specification_field_trace>].
was thrown during the creation of a connection
to messaging engine xyzNode01.server1-xyz on bus xyz.
```

By default, the bus-enabled web services component can connect to a secure bus destination through the service integration resource adapter. Therefore the configuration must have been changed in some way.

The default configuration that the bus-enabled web services component uses to access a secure bus is as follows:

- Access to a bus is configured through the *bus connector role*. By default, every bus connector role includes a group called *server*. Members of this group are authorized to connect to the bus.
- The service integration resource adapter uses a J2C activation specification to communicate with the bus. By default, this activation specification has a Boolean custom property **useServerSubject** that is set to true. This property allows the service integration resource adapter to connect to the bus as a subject (a member) of the server group.

You can override this default configuration by defining an authentication alias that the service integration resource adapter uses to access the bus. For more information, see [Overriding the default security configuration between bus-enabled web services and a secure bus](#).

For detailed information about the default configuration, and the effect of modifying or overriding this configuration, see [Bus-enabled web services default configuration for accessing a secure bus](#).

To help you narrow down the problem area, set the application server trace string to `com.ibm.ws.sib.webservices.*=all=enabled` because there are several components that can be the cause of the problem. In the trace, check `SibRaMessagingEngineConnection.createConnection()`:

- The phrase `Creating connection with Userid and password` indicates that the system has found, and is attempting to use, an authentication alias configured for the J2C activation specification.
- The phrase `No userid/password passed. Creating connection using server subject` indicates that the system has found, and is attempting to use, the server subject.

The service integration bus times out while an outbound service is waiting for the response from a target service

The following error can occur in the service integration bus when an outbound service is waiting for a response from a target web service:

```
java.net.SocketTimeoutException: Socket operation timed out before it could be completed
```

The default timeout value is 60 seconds, so this error occurs if the target web service takes longer than 60 seconds to respond. You can increase the timeout value by setting a **timeout** custom property on the inbound port as described in [Inbound Ports \[Settings\]](#).

You must manually install one of the bus-enabled web services applications or resources

The following bus-enabled web services applications and resources are installed automatically when they are first needed:

- The bus-enabled web services application (the application that lets you configure and access web services through a service integration bus).
- The service integration technologies resource adapter (used to invoke web services at outbound ports).
- The endpoint listener applications (used to enable the points at which messages for inbound services are received).

For example, an endpoint listener application is installed as part of the process of creating a new endpoint listener configuration.

However if you do have to manually install one of these applications, you can do so by using the supplied `sibwsInstall.jacl` script, and following the instructions given in the [WebSphere Application Server Version 6.0.x topic: Installing the bus-enabled web services applications and resources](#)

You have a client application that works under WebSphere Application Server Version 5.1, but in later versions you get problems caused by poorly-formed requests or responses

Bus-enabled web services check the validity of web service messages more thoroughly than is done in WebSphere Application Server Version 5.1. As a result, some client applications that use poorly-formed requests or responses (where the message parts are misnamed), and that work when using Version 5.1, are identified as poorly-formed in later versions. For the steps to take to resolve the problem, see [“Toleration of poorly-formed SOAP messages”](#) on page 290

A JAX-RPC client running on WebSphere Application Server Version 5.1 uses SOAP over JMS to invoke a web service running on a Version 5 application server. No user ID or password is required on the target MQ Series queue. After the application server is migrated to a later version, and to use default messaging, client requests fail because basic authentication is now enabled

The problem appears as a log message:

```
SibMessage W [:] CWSIT0009W: A client request failed in the
application server with endpoint <endpoint_name> in bus your_bus
with reason: CWSIT0016E: The user ID null failed authentication
in bus your_bus.
```

For the steps to take to resolve the problem, see the following service integration technologies troubleshooting tip: “After you migrate a Version 5.1 application server to WebSphere Application Server Version 7.0 or later, existing web services clients can no longer use SOAP over JMS to access services hosted on the migrated server.” on page 209

You unsuccessfully attempt to create an Informix database for use with the SDO repository, and receive the message “No Transaction Isolation on non-logging databases.”

If “No Transaction Isolation on non-logging databases.” is displayed as part of an exception message, it is because logging is disabled on the Informix database being used. To use an Informix database with the SDO repository, you must enable logging.

The full exception text is:

```
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:
PMGR1013E: Exception occurred when verifying current backend id
INFORMIX_V94: javax.resource.spi.ResourceAllocationException:
DSRA0080E: An exception was received by the Data Store Adapter.
See original exception message:
No Transaction Isolation on non-logging dbs., error code:
DSA_ERROR, error code: DSA_ERROR vmcid: 0x0 minor code: 0
completed: No
```

Logging is disabled by default, so the create database statement CREATE DATABASE SDOREP; results in an exception at run time. When you create the database, use one of the following database creation statements:

- CREATE DATABASE SDOREP WITH LOG;
- CREATE DATABASE SDOREP WITH BUFFERED LOG;
- CREATE DATABASE SDOREP WITH LOG MODE ANSI;

You have an inbound service that, according to the administrative console, is published to a UDDI registry. When you check the UDDI registry you discover that the service is not listed there. When you attempt to use the administrative console to republish the service to UDDI, the attempt is unsuccessful

The service was published to the UDDI registry, and the service configuration shown in the WebSphere Application Server administrative console includes a UDDI Service Key, but the service has subsequently been unpublished from UDDI without the corresponding update being applied to the WebSphere Application Server master configuration. One way in which this situation can arise is if you use the administrative console to delete an inbound service that is published to a UDDI registry, then log out of the

administrative console without saving your changes. In this case, the service is unpublished from the UDDI registry, but not deleted from WebSphere Application Server (because the delete request is not confirmed, and therefore is not applied).

To update the service configuration information and republish the service to UDDI, use the administrative console to complete the following steps:

1. In the navigation pane, click **Service integration -> Buses -> bus_name -> [Services] Inbound Services -> service_name**. The current settings for this inbound service are displayed.
2. Click **Reload template WSDL**, then save the changes.
3. Click **Unpublish from UDDI**, then save the changes.
4. Click **Publish to UDDI**, then save the changes.

The service is successfully republished, and is reinstated in the UDDI registry.

If the bus needs to pass messages through an authenticating proxy server to retrieve WSDL documents, then you must use command-line tools to retrieve the WSDL

Neither the administrative console panels used to create a new web service configuration, nor the **Reload WSDL** button provided on the panels used to modify an existing web service configuration, allow you to enter a J2C authentication alias for WSDL retrieval. Therefore when you create or modify inbound and outbound services, if the bus needs to pass messages through an authenticating proxy server to retrieve WSDL documents then you must use one of the following command-line tools to retrieve the WSDL:

- Creating a new inbound service configuration by using the wsadmin tool.
- Creating a new outbound service configuration by using the wsadmin tool.
- Refreshing the inbound service WSDL file by using the wsadmin tool.
- Refreshing the outbound service WSDL file by using the wsadmin tool.

If you are using JMS to connect to a remote bus, extra configuration is required to allow web service clients to connect to the bus

A web service client application running in a server that is a member of a bus can locate a messaging engine in that bus. A web service client application running outside of an application server - for example, running outside the WebSphere Application Server environment - cannot locate directly a suitable messaging engine to connect to in the target bus. Similarly, a web service client application running on a server in one cell that needs to connect to a target bus in another cell cannot locate directly a suitable messaging engine to connect to in the target bus.

To enable the web service client application to contact a target messaging engine in a remote bus, configure the JMS connection factory that the client uses so that the client can connect to a bootstrap messaging engine in the remote bus. The bootstrap messaging engine then identifies the target engine, and the information required to access the target engine is passed back to the client. For the bootstrap process to be possible, configure one or more provider end points in the connection factory used by the client. For more information, see *Configuring connection to a non-default bootstrap server*.

You pass a large attachment through the service integration bus and you get an out-of-memory error in the Java virtual machine

If this error occurs, increase the heap size as described in *Tuning bus-enabled web services*.

When you try to create a new endpoint listener configuration by using the administrative console, and click New on the endpoint listener collection panel, the “Please wait” icon is displayed but the target panel does not appear

This problem is only found with older versions of the Mozilla web browser. Upgrade your browser to Version 1.4 or later. As a workaround, click **New** a second time and the target panel is displayed.

When you try to create a new inbound service through the administrative console, the drop-down list of destinations is empty and the wizard stops you at step 1 with the error message “A destination must be selected”

This can only happen if there is no messaging engine on the service integration bus on which you are creating your inbound service. Create a messaging engine, then create a service destination, then re-run the wizard to create a new inbound service configuration.

You experience problems with handling SOAP messages with attachments

See You pass a large attachment through the service integration bus and you get an out-of-memory error in the Java virtual machine..

You are trying to send a SOAP over HTTPS message, and you are receiving a Malformed URLError exception error

The service integration technologies can use Secure Sockets Layers (SSL) to invoke external web services that include https:// in their addresses. For more information see Invoking outbound services over HTTPS.

You get JNDI lookup errors when you use the same names for JMS messaging queues and queue connection factories that run on application servers on different machines

You should not use the same names for messaging queues and queue connection factories that run on application servers on different machines, because the service integration technologies always look first for JMS destinations locally, and only use the full JNDI reference if they cannot find the destination locally. If you configure as an outbound service a web service that is hosted on a remote machine, and you use the same names for messaging queues and queue connection factories on the remote machine and on the machine on which the outbound service is hosted, then the service integration technologies find and use the local queues even if the remote JNDI destination is provided in full in the WSDL service definition.

You are password-protecting a web service operation, but when you install the sibwsauthbean.ear file, an error message is displayed in the WebSphere Application Server administrative console detailing a Java Naming and Directory Interface (JNDI) problem

When you password-protect a web service operation, check that you enter, in the “EJB References” for the authorization session bean, the correct JNDI name of the imported web service enterprise bean. Note that this home is case sensitive.

You are getting SOAP fault messages, but cannot determine the precise problem from the fault message

If you receive a SOAP fault message with a faultstring that is just the value of one of the parameters of the invocation, that means the parameter value is not valid. For example if you have a service that expects an int parameter and you send it a message containing the value “1.1”, then the fault message you receive contains 1.1 as the fault string:

```
<faultcode>SOAP-ENV:Client</faultcode>  
<faultstring>1.1</faultstring>
```

This message is consistent with Apache SOAP behavior, and is not correctable by bus-enabled web services.

You get listener port timeout errors when large messages are passed by using the SOAP over JMS endpoint listener

As with any synchronous endpoint listener, timeout errors can occur. To minimize the frequency of timeout errors, increase the timeout settings for the endpoint listener. If the problem persists, then disable trace and logging for service integration technologies by setting the application server trace string to `com.ibm.ws.sib.webservices.*=all=disabled`.

Bus-enabled web services: Known restrictions

There are a small number of known restrictions that apply when using service integration bus-enabled web services.

SOAP restrictions

- “Passing SOAP headers through the service integration bus”
- “Limitations in the support for SOAP with attachments”
- “Toleration of poorly-formed SOAP messages” on page 290

Security restrictions

- “JAAS Subject credential tokens not guaranteed to be available on outbound services” on page 290
- “Limitations in support for previous WS-Security draft specifications” on page 291

Other restrictions

- “Limitations regarding the Java types used by services that are retargeted through a JAX-RPC client application” on page 291
- “When you configure an outbound service to use a WSDL port that uses the EJB binding, all the classes passed to the enterprise bean must be present in the WebSphere Application Server class path” on page 291

Passing SOAP headers through the service integration bus

If the WSDL for your service contains `<soap:header>` elements within the `<wsdl:definition>` element, then the bus passes the SOAP headers through. This behavior is correct. However, you also see the following effects:

- The SOAP headers are not included in the WSDL that the service integration technologies generates.
- If you set the “must understand” flag on the SOAP message, then you get an error message.

Limitations in the support for SOAP with attachments

The service integration bus supports SOAP messages that contain either old-style attachments (as described in the SOAP with attachments W3C Note) or attachments that use the Web Services-Interoperability (WS-I) Attachments Profile Version 1.0. If you have to transform attachments from one style to another, you can use a mediation to map between attachment encoding styles.

Bus-enabled web services cannot invoke a web service that is hosted by WebSphere Application Server if the service has an operation that does not have attachments in its request message and does return an attachment in its response message.

The following scenarios are also not supported:

- Using DIME.
- Using the `mime:mimeXml` WSDL tag.
- Nesting a `mime:multipartRelated` inside a `mime:part`.
- Using arrays or vectors of `DataHandlers`, images, and so on.

The MIME headers from the incoming message are not preserved for referenced attachments. The outgoing message contains new MIME headers for `Content-Type`, `Content-Id` and `Content-Transfer-Encoding`.

If you pass a large attachment through the service integration bus, you might get an out-of-memory error in the Java virtual machine. To solve this problem, increase the JVM heap size as described in [Tuning bus-enabled web services](#).

For more information, see [Passing SOAP messages with attachments through the service integration bus](#).

JAAS Subject credential tokens not guaranteed to be available on outbound services

When using WS-Security, the following contents of a JAAS Subject credentials set are not guaranteed to be available to code running on an outbound service if they are set on the processing of an inbound service request:

- Non-serializable contents.
- Any tokens that implement `com.ibm.wsspi.security.token.Token` or any of its sub-interfaces, and that do not set the **forwardability** attribute to true.

For example, if a custom `TokenConsumer` is configured within the WS-Security configuration and bindings applied to an inbound port, and the `TokenConsumer` sets a token within the private credentials of the JAAS subject, and that token implements `com.ibm.wsspi.security.token.Token` and sets the **forwardability** attribute to false, then any custom `TokenGenerator` configured on the corresponding outbound port WS-Security configuration and bindings should not rely on that token being available within the JAAS Subject.

Toleration of poorly-formed SOAP messages

Bus-enabled web services check the validity of web service messages more thoroughly than is done in WebSphere Application Server Version 5.1. As a result, some client applications that use poorly-formed requests or responses (where the message parts are misnamed), and that work when using Version 5.1, are identified as poorly-formed in later versions.

Bus-enabled web services support applications that produce messages where the message parts are misnamed, provided that they still match the general form of the schema. With this support:

- Poorly-formed messages can be accepted by the bus.
- Poorly-formed messages can be produced by the bus.

For output messages, a poorly formed message is only produced if the input message is poorly formed and the message does not have to be rewritten by bus-enabled web services. Whenever bus-enabled web services have to rewrite the message (for example because it has been modified by a mediation) they produce a well-formed SOAP message using the correct names for the parts as defined in the WSDL document. In each of these cases, if your service or client relies on the response message part names being misnamed, either modify the client or restructure the WSDL that is associated with the bus-enabled web service so that the part names match those that the applications are expecting.

Note: Only incorrect part names are tolerated. Incorrect operation names or incorrect part structure are not tolerated.

Limitations in support for previous WS-Security draft specifications

Versions of the WS-Security draft specification that were supported by the general web services support in previous versions of WebSphere Application Server are not supported by service integration technologies. Service integration technologies only support the “OASIS Web Services Security Version 1.0 specification,” the “Username token Version 1.0 profile,” and the “X.509 token Version 1.0 profile.” For more information about these supported specifications and profiles, see Supported functionality from OASIS specifications.

All client applications and target services that use WS-Security to interact with service integration technologies must also conform to the supported levels of these specifications. Client applications and target services that conform to previously supported versions of the WS-Security draft specification are not able to interact with service integration technologies because the wire format of the SOAP message with WS-Security has changed in the OASIS Web Services Security Version 1.0 specification and is not compatible with previous drafts of the specification.

Limitations regarding the Java types used by services that are retargeted through a JAX-RPC client application

When you pass messages into the service integration bus at a destination by sending web service messages directly over the bus from a JAX-RPC client, there are limitations regarding the Java types you can use.

You can only retarget services that limit the types that are used in their interface to those that have defined mappings in the JAX-RPC specification. This limits the support to a subset of the possible XML schema that can be used in a WSDL document. For example, if the interface has any element that maps to `SOAPElement` it cannot be retargeted over the bus.

When you configure an outbound service to use a WSDL port that uses the EJB binding, all the classes passed to the enterprise bean must be present in the WebSphere Application Server class path

When you configure an outbound service to use a WSDL port that uses the EJB binding, service integration technologies invokes the service using Remote Method Invocation over Internet Inter-ORB Protocol (RMI-IIOP). However, all the classes passed to the enterprise bean must be present in the WebSphere Application Server class path. For example:

- If you pass an object of type `Address`, that class and the classes of all the objects serialized within an `Address` object must be present on the WebSphere Application Server class path.
- If the signature of a method on the enterprise bean contains a `java.util.List` object and it is expected that the list is a list of `Address` objects then the `Address` class must be on the WebSphere Application Server class path.

Web services gateway troubleshooting tips

Use this set of specific tips to help you troubleshoot problems you experience when using the web services gateway.

To help you identify and resolve web services gateway-related problems, use the WebSphere Application Server trace and logging facilities as described in `ttrb_setUpCTRACE.html`.

To enable trace for the gateway, set the application server trace string to `com.ibm.ws.sib.webservices.*=all=enabled:com.ibm.ws.wsgw.*=all=enabled`. If you encounter a problem that you think might be related to the gateway, 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.

The web services gateway is implemented as an administrative layer on top of service integration bus-enabled web services. Therefore most known problems that you might experience when using the gateway are actually bus-enabled web services-related problems, and are documented in Bus-enabled web services troubleshooting tips.

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 web services gateway component is CWWSG and the prefix for the bus-enabled web services component is CWSWS.

The Troubleshooter reference: Messages topic contains information about the gateway and bus-enabled web services 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:

- “When you pass messages directly to a bus destination, the default RPC-encoded web services string array message does not interoperate successfully with some target service providers.”
- “A SOAP over JMS web service message is sent by the web services gateway as a `JmsBytesMessage`, rather than a `JmsTextMessage`.” on page 293
- “If you create a gateway service, its WSDL description is not created in the SDO repository until you attempt to access the gateway service WSDL file through a web browser.” on page 293
- “You migrate a gateway from WebSphere Application Server Version 5.1 to a later version. When you try to access a gateway service, your client receives one or more error messages stating No response body is available, or Null response message, or The message body did not match any of the definitions in the WSDL.” on page 294
- “You choose to generate a web service client from the WSDL for the target service, rather than the gateway service. When you try to access the target service through the gateway, your client receives the following error message: CWSIF0304E: The message body did not match any of the definitions in the WSDL.” on page 294
- “You migrate a gateway that contains filters from WebSphere Application Server Version 5.1 to a later version, and your filters no longer work.” on page 294
- “When you migrated a gateway that contains filters from WebSphere Application Server Version 5.1 to Version 7.0 or later, destinations with names ending `StorageQueue` were created. These destinations are not removed automatically if you subsequently delete the migrated gateway.” on page 295
- “You have a client application that works under WebSphere Application Server Version 5.1, but under later versions you get problems caused by poorly-formed requests or responses.” on page 295
- “A JAX-RPC client running on WebSphere Application Server Version 5.1 uses SOAP over JMS to invoke a web service running on a Version 5.1 application server. No user ID or password is required on the target MQ Series queue. After the application server is migrated to a later version, and to use default messaging, client requests fail because basic authentication is now enabled.” on page 295
- “The gateway panels in the administrative console are only available in WebSphere Application Server Network Deployment if you are working with a deployment manager profile.” on page 295

When you pass messages directly to a bus destination, the default RPC-encoded web services string array message does not interoperate successfully with some target service providers.

You can pass messages directly to a bus destination by overriding the JAX-RPC client binding namespace and endpoint address. However:

- The default RPC-encoded web services string array message that is generated might not interoperate successfully with some target service providers.
- The string array message produced is not exactly the same as the standard JAX-RPC equivalent, which can interoperate successfully.

Here are examples of the two different messages:

- Service integration bus message:

```
<partname env:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/  
  xsi:type='ns1:ArrayOf_xsd_string'  
  <item xsi:type='xsd:anySimpleType'>namevalue</item>  
</partname>
```

- JAX-RPC client message:

```
<partname xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]">  
  <item>namevalue</item>  
</partname>
```

To modify the default behavior to send a string array message that is fully compatible with standard JAX-RPC, set the following JVM custom property. Setting this property modifies the default behavior for all outbound JMS web services invocations sent from the bus:

1. Start the administrative console.
2. Navigate to **Servers -> Server Types -> WebSphere application servers -> server_name -> [Server Infrastructure] Java and Process Management -> Process Definition > [Additional Properties] Java Virtual Machine -> [Additional Properties] Custom Properties** then click **New**.
3. Create the following JVM custom property. Note that the values shown are case sensitive.
 - Name: com.ibm.websphere.sib.webservices.useTypeSoapArray
 - Value: true
4. Restart the application server.

A SOAP over JMS web service message is sent by the web services gateway as a JmsBytesMessage, rather than a JmsTextMessage.

By default on WebSphere Application Server Version 6 or later, a SOAP over JMS web service message sent by the web services gateway is sent as a JmsBytesMessage, whereas on WebSphere Application Server Version 5.1 the web services gateway sends a JmsTextMessage.

To modify the default behavior to send a compatible JmsTextMessage, set the following JVM custom property. Setting this property modifies the default behavior for all outbound JMS web services invocations sent from the bus:

1. Start the administrative console.
2. Navigate to **Servers -> Server Types -> WebSphere application servers -> server_name -> [Server Infrastructure] Java and Process Management -> Process Definition > [Additional Properties] Java Virtual Machine -> [Additional Properties] Custom Properties** then click **New**.
3. Create the following JVM custom property. Note that the values shown are case sensitive.
 - Name: com.ibm.ws.sib.webservices.useSOAPJMSTextMessages
 - Value: true
4. Restart the application server.

If you create a gateway service, its WSDL description is not created in the SDO repository until you attempt to access the gateway service WSDL file through a web browser.

Every gateway service has an associated inbound service. The gateway service WSDL file is associated with this inbound service and should only be needed if the message originates from an inbound service. Therefore the WSDL description is created in the SDO repository the first time it is called by an inbound service or through a web browser.

If your applications are posting messages into the bus from sources other than the inbound service (for example if you are using mediation handlers to manipulate SDO messages to or from a gateway service),

the applications should use either the WSDL associated with the target (outbound) service or some other compatible WSDL.

You migrate a gateway from WebSphere Application Server Version 5.1 to a later version. When you try to access a gateway service, your client receives one or more error messages stating “No response body is available”, or “Null response message”, or “The message body did not match any of the definitions in the WSDL”.

When the Version 5 gateway generates WSDL for a gateway service, the generated namespace contains the service name. For example: `namespace="http://griddev:9080/wsgw#yourService"`. However in later versions the generated WSDL for a gateway service does not contain the service name. For example: `namespace="http://griddev:9080/wsgw"`.

If your WSDL binding and encoding style is document literal then your clients still work with the migrated gateway, because document literal style does not use the namespace attribute. However, if you used the Version 5.1 gateway service WSDL to generate your web service clients and your WSDL binding and encoding style is not document literal, then after migration you must regenerate the client stubs by using the new gateway service WSDL.

You choose to generate a web service client from the WSDL for the target service, rather than the gateway service. When you try to access the target service through the gateway, your client receives the following error message: “CWSIF0304E: The message body did not match any of the definitions in the WSDL”.

The benefit of choosing to generate a web service client from the WSDL for the target service is that you can move between going to the target service directly or going through the gateway to the target service just by changing the URL in the generated stubs. To enable this approach in this version of the gateway, use the administrative console to set the custom property `com.ibm.websphere.wsgw.mapSoapBodyNamespace` to `false` on each inbound service that is associated with a gateway service.

Note: This approach is not flagged by default as a possible error in WebSphere Application Server Version 5.1, so you might encounter this problem for the first time when you migrate a gateway configuration from a Version 5.1 application server to the gateway capability on an application server on a later version. After migration, you have two choices:

- If you want to retain the flexibility to reroute your messages by changing the URL in the generated stubs, set the custom property `com.ibm.websphere.wsgw.mapSoapBodyNamespace` to `false` as described above.
- If you no longer need this flexibility, regenerate the client stubs by using the gateway service WSDL of the later version.

You migrate a gateway that contains filters from WebSphere Application Server Version 5.1 to a later version, and your filters no longer work.

The use of filters was deprecated in Version 5.1.1 and support for filters was removed in Version 7.0. The role formerly played by filters is now undertaken by a combination of JAX-RPC handlers and service integration bus mediations. If you migrate a web services gateway that includes a routing filter, you can recreate the filter functions as described in Choosing a target service and port through a routing mediation.

When you migrated a gateway that contains filters from WebSphere Application Server Version 5.1 to Version 7.0 or later, destinations with names ending “StorageQueue” were created. These destinations are not removed automatically if you subsequently delete the migrated gateway.

These additional destinations were required to support existing Version 5.1 gateway filters in the Version 6 environment. Gateway filters are no longer supported, and so these “StorageQueue” destinations are no longer needed in Version 7.0 or later. However they are not removed automatically when you migrate to Version 7.0 or later or you delete the gateway instance.

During migration to Version 6, these additional destinations were created in one of the following two ways:

- If you specified the `-Q` parameter (shared filter correlation queue name), then a single queue destination of the specified name was created (if it did not already exist) and shared by all gateway services that had associated filters.
- If you did not specify the `-Q` parameter, then a separate queue destination was created for each gateway service that had associated filters. Each of these destinations has a name derived from the name of the corresponding gateway service request and reply destinations, and ending in `StorageQueue`.

After you delete the gateway instance, identify each associated storage queue destination by name, and delete it as described in [Deleting a non-topic space bus destination](#).

You have a client application that works under WebSphere Application Server Version 5.1, but under later versions you get problems caused by poorly-formed requests or responses.

Bus-enabled web services check the validity of web service messages more thoroughly than is done in WebSphere Application Server Version 5.1. As a result, some client applications that use poorly-formed requests or responses (where the message parts are misnamed), and that work when using Version 5.1, are identified as poorly-formed in later versions. For the steps to take to resolve the problem, see “Toleration of poorly-formed SOAP messages” on page 290

A JAX-RPC client running on WebSphere Application Server Version 5.1 uses SOAP over JMS to invoke a web service running on a Version 5.1 application server. No user ID or password is required on the target MQ Series queue. After the application server is migrated to a later version, and to use default messaging, client requests fail because basic authentication is now enabled.

The problem appears as a log message:

```
SibMessage W [:] CWSIT0009W: A client request failed in the application
server with endpoint <endpoint_name> in bus your_bus with reason:
CWSIT0016E: The user ID null failed authentication in bus your_bus.
```

For the steps to take to resolve the problem, see the following service integration technologies troubleshooting tip: “After you migrate a Version 5.1 application server to WebSphere Application Server Version 7.0 or later, existing web services clients can no longer use SOAP over JMS to access services hosted on the migrated server.” on page 209

The gateway panels in the administrative console are only available in WebSphere Application Server Network Deployment if you are working with a deployment manager profile.

The panels are not available in a WebSphere Application Server Network Deployment stand-alone server profile. However, the `wsadmin` command scripts that provide equivalent functions by using the `wsadmin` tool are available for both the stand-alone and deployment manager profiles. The gateway administrative commands cover all the main requirements, except for creating a new gateway instance. You can create a

gateway instance and its associated default proxy WSDL location information through the wsadmin scripting client by using Jacl. Here is an example script:

- Using Jython:

```
wsgwAttribs = []
wsgwAttribs.append(["name", wsgwName])
wsgwAttribs.append(["wsdlServiceNamespace", wsgwNamespace])
wsgw = AdminConfig.create("WSGWInstance", bus, wsgwAttribs )
wsgwWsd1Attribs = []
wsgwWsd1Attribs.append(["WSDLLocation", wsgwWsd1Location])
AdminConfig.create("SIBWSWSDLLocation", wsgw, wsgwWsd1Attribs,
"defaultProxyWSDLLocation" )
```

- Using Jacl:

```
set wsgwAttribs {}
lappend wsgwAttribs [list "name" $wsgwName]
lappend wsgwAttribs [list "wsdlServiceNamespace" $wsgwNamespace]
set wsgw [$AdminConfig create "WSGWInstance" $bus $wsgwAttribs]
set wsgwWsd1Attribs {}
lappend wsgwWsd1Attribs [list "WSDLLocation" $wsgwWsd1Location]
$AdminConfig create "SIBWSWSDLLocation" $wsgw $wsgwWsd1Attribs
"defaultProxyWSDLLocation"
```

In this example, bus or \$bus is the ID of the service integration bus in which the gateway instance is created. For more information about the WebSphere Application Server AdminConfig object, enter “AdminConfig.help()” from the wsadmin command line.

WS-Notification troubleshooting tips

Tips for troubleshooting WS-Notification publish and subscribe messaging for web services.

To help you identify and resolve WS-Notification problems, use the WebSphere Application Server trace and logging facilities as described in [Setting up component trace \(CTRACE\)](#).

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.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

A list of the main known restrictions that apply when using WS-Notification is provided in “WS-Notification: Known restrictions” on page 306.

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:

- “A JAX-WS application that is a raw consumer of brokered notifications must recognize a notification broker SOAP action”
- “The PublisherRegistrationManager.wsdl file is not successfully parsed by wsimport unless you include a JAX-WS bindings file” on page 298
- “A Version 7.0 WS-Notification service receives a triggerActionNotSupportedFault when using operations on the PausableSubscriptionManager interface of a JAX-WS demand-based publisher” on page 299
- “There are Connection timed out errors in your broker server logs” on page 300
- “There are Out of memory errors in your broker server logs” on page 300
- “WebSphere Application Server Version 6.1 client applications must handle the additional error conditions introduced in the final Version 1.3 WS-Notification standards” on page 300
- “An exception occurs because the SDO repository is not configured correctly” on page 300
- “Multiple messages are received by a notification consumer for each event notification” on page 301
- “Problems can occur when deleting administered subscribers and messaging engines” on page 301
- “There are differences when using bus destinations with Version 6.1 WS-Notification services” on page 302
- “An inbound (application to broker) notification is not successful” on page 302
- “An outbound (broker to application) notification is not successful” on page 303
- “How to tidy up stateful resources that are not automatically deleted” on page 304
- “A durable subscription that was created by WS-Notification cannot be deleted when using the service integration bus panel” on page 304
- “Administrative stop of a messaging engine” on page 304
- “Failures as a result of changes in topic space and topic namespace configurations” on page 305
- “You cannot create a Version 6.1 WS-Notification service without a configured SDO repository” on page 305
- “An exception occurs when a JAX-WS client that does not have internet access attempts to contact a WS-Notification service” on page 306

A JAX-WS application that is a raw consumer of brokered notifications must recognize a notification broker SOAP action

JAX-WS supports action-based dispatch, and JAX-WS raw consumer applications must accept the <http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify> action URI. You can enable this in any of the following ways:

- In the raw consumer application WSDL file, modify the SOAP binding information to contain the notify action URI:

```
<wsdl:operation name="oneWayRawSubscriptionNotify">
  <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify" />
  <wsdl:input name="oneWayRawSubscriptionNotifyRequest">
    <soap:body use="literal" />
  </wsdl:input>
</wsdl:operation>
```
- In the raw consumer application WSDL file, modify the port type information to contain the notify action URI:

```
<wsdl:operation name="oneWayRawSubscriptionNotify">
  <wsdl:input message="impl:oneWayRawSubscriptionNotifyRequest"
    name="oneWayRawSubscriptionNotifyRequest"
    wsam:Action="http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify">
  </wsdl:input>
</wsdl:operation>
```
- Use a JAX-WS annotation to specify the action URI <http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify> in the application code. For more information, see the **action** property of the **WebMethod** annotation in topic JAX-WS annotations .

The PublisherRegistrationManager.wsdl file is not successfully parsed by wsimport unless you include a JAX-WS bindings file

When you publish the WSDL files for a WS-Notification application to a compressed file, then run the wsimport command against the PublisherRegistrationManager.wsdl file, the following fault message is displayed:

```
[ERROR] the following naming conflicts occurred:  
com.ibm.websphere.wsn.publisher_registration_manager.ResourceNotDestroyedFault  
line 2 of file:/path_to_wsdl/PublisherRegistrationManager.wsdl
```

This fault occurs because the WSDL for the publisher registration manager uses both the WS-Notification and the WS-ResourceLifetime specifications; both of which refer to a **ResourceNotDestroyedFault** element that shares the same message name. Here is the relevant part of the publisher registration manager WSDL file:

```
<wsdl:operation name="DestroyRegistration">  
  <wsdl:input name="DestroyRegistrationRequest" message="wsn-brw:DestroyRegistrationRequest"  
    wsam:Action="http://docs.oasis-open.org/wsn/brw-2/PublisherRegistrationManager/DestroyRegistrationRequest"  
    wsaw:Action="http://docs.oasis-open.org/wsn/brw-2/PublisherRegistrationManager/DestroyRegistrationRequest"/>  
  <wsdl:output name="DestroyRegistrationResponse" message="wsn-brw:DestroyRegistrationResponse"  
    wsam:Action="http://docs.oasis-open.org/wsn/brw-2/PublisherRegistrationManager/DestroyRegistrationResponse"  
    wsaw:Action="http://docs.oasis-open.org/wsn/brw-2/PublisherRegistrationManager/DestroyRegistrationResponse"/>  
  <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"  
    wsam:Action="http://docs.oasis-open.org/wsrf/fault"  
    wsaw:Action="http://docs.oasis-open.org/wsrf/fault"/>  
  <wsdl:fault name="ResourceNotDestroyedFault" message="wsn-brw:ResourceNotDestroyedFault"  
    wsam:Action="http://docs.oasis-open.org/wsn/fault" wsaw:Action="http://docs.oasis-open.org/wsn/fault"/>  
</wsdl:operation>  
  
<!-- Some parts have been omitted -->  
  
<!-- An extract from WS-ResourceLifetime -->  
<wsdl:operation name="Destroy">  
  <wsdl:input name="DestroyRequest" message="wsrf-rlw:DestroyRequest"  
    wsam:Action="http://docs.oasis-open.org/wsrf/rlw-2/ImmediateResourceTermination/DestroyRequest"  
    wsaw:Action="http://docs.oasis-open.org/wsrf/rlw-2/ImmediateResourceTermination/DestroyRequest"/>  
  <wsdl:output name="DestroyResponse" message="wsrf-rlw:DestroyResponse"  
    wsam:Action="http://docs.oasis-open.org/wsrf/rlw-2/ImmediateResourceTermination/DestroyResponse"  
    wsaw:Action="http://docs.oasis-open.org/wsrf/rlw-2/ImmediateResourceTermination/DestroyResponse"/>  
  <wsdl:fault name="ResourceNotDestroyedFault" message="wsrf-rlw:ResourceNotDestroyedFault"  
    wsam:Action="http://docs.oasis-open.org/wsrf/fault"  
    wsaw:Action="http://docs.oasis-open.org/wsrf/fault"/>  
  <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"  
    wsam:Action="http://docs.oasis-open.org/wsrf/fault"  
    wsaw:Action="http://docs.oasis-open.org/wsrf/fault"/>  
  <wsdl:fault name="ResourceUnavailableFault" message="wsrf-rw:ResourceUnavailableFault"  
    wsam:Action="http://docs.oasis-open.org/wsrf/fault"  
    wsaw:Action="http://docs.oasis-open.org/wsrf/fault"/>  
</wsdl:operation>
```

To resolve this naming conflict, you must include the JAX-WS bindings file `ibm-wsn-jaxws.xml` as an argument to the wsimport command. This bindings file ensures that the conflicting elements are mapped to different class names.

The `ibm-wsn-jaxws.xml` file is located in the `app_server_root/util` directory. For example: `c:\was\util\ibm-wsn-jaxws.xml`. This bindings file expects to find the WSDL file to which it refers in the same directory as itself, so before you run the wsimport command you must copy the bindings file to the directory that holds your `PublisherRegistrationManager.wsdl` file. Here is an example of how to run the wsimport command to include the `ibm-wsn-jaxws.xml` file:

```
c:\was\bin\wsimport -b ibm-wsn-jaxws.xml -keep PublisherRegistrationManager.wsdl
```

A Version 7.0 WS-Notification service receives a triggerActionNotSupportedFault when using operations on the PausableSubscriptionManager interface of a JAX-WS demand-based publisher

You have a JAX-WS demand-based publisher registered with a Version 7.0 type of WS-Notification service. When the service invokes any of the operations on the PausableSubscriptionManager interface of the publisher, the publisher responds with a triggerActionNotSupportedFault exception message. The operations that trigger this message are Renew, Unsubscribe, PauseSubscription or ResumeSubscription.

You see messages similar to the following text in the SystemOut.log file for the server. In this example the fault message is triggered in response to the broker attempting to unsubscribe from the demand-based publisher.

```
triggerActionNotSupportedFault triggerActionNotSupportedFault: messageContext:
[MessageContext: logID=urn:uuid:13616A3EB4F278A3DC1221827497002] problemAction:
http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeRequest
```

```
CWSJN1029W: An attempt was made to perform an operation on a remote NotificationProducer
  located with endpoint Address[[xxx/prod_service/NotificationProducerPort]],
ReferenceParams[] but the operation could not be completed. This operation will be
retried automatically in 2 seconds.
The operation failed due to com.ibm.ws.sib.wsn.webservices.WSNWSEException:
```

```
CWSJN5035E: An internal error occurred: Unable to unsubscribe from remote publisher
at endpoint reference Address[[xxx/prod_service/PausableSubscriptionManagerPort]],
ReferenceParams[] due to javax.xml.ws.soap.SOAPFaultException:
The [action] cannot be processed at the receiver.
```

The server continues to unsuccessfully attempt to unsubscribe from the publisher at ever-increasing time intervals.

When the WSDL file for your demand-based publisher does not specify a SOAP action pattern, WS-Addressing generates one by default. If your publisher uses the PausableSubscriptionManager port type for its binding, the default action patterns generated by WS-Addressing do not match the actions defined by the WS-Notification specification.

Note: This problem only occurs if your publisher uses the PausableSubscriptionManager port type. If your publisher uses the SubscriptionManager port type, then the default action patterns generated by WS-Addressing match those in the WS-Notification specification.

To resolve this problem, in the WSDL file for your demand-based publisher you must explicitly specify the SOAP action to use for each of the operations on the PausableSubscriptionManager interface. The action URI to use for each operation is defined in the Web Services Base Notification 1.3 (WS-BaseNotification) specification available from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn For example, the WS-Addressing action for the Unsubscribe operation is defined in the specification as <http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeRequest>, therefore you must use this action for the Unsubscribe operation in your publisher WSDL file. Here is an excerpt from the binding section of such a WSDL file:

```
<wsdl:binding name="PausableSubscriptionManagerBinding" type="wsn-bw:PausableSubscriptionManager">
<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />

<wsdl:operation name="Unsubscribe">
<soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeRequest" />
```

Similarly you must specify the corresponding actions for the Renew, PauseSubscription and ResumeSubscription operations in your publisher WSDL file.

There are “Connection timed out” errors in your broker server logs

If you see `WebServicesFault("Connection timed out")` errors in your WS-Notification broker server logs, and you have a large number of consumers subscribed to your WS-Notification service, the broker might have exceeded the maximum number of connections in the HTTP outbound connector connection pool when sending outbound notification messages to subscribed consumers.

To increase the number of outbound HTTP connections in the pool, set the **`com.ibm.websphere.webservices.http.maxConnection`** custom property on the server or servers on which your broker is running. For more information about this property, see HTTP transport custom properties for web services applications.

There are “Out of memory” errors in your broker server logs

If you see “Out of memory” errors in your WS-Notification broker server logs, and you have a large number of consumers subscribed to your WS-Notification service, the broker might have exceeded the available JVM heap size allocated to the server on which it is running.

To increase the maximum heap size for the server or servers on which your broker is running, see Tuning the IBM virtual machine for Java.

WebSphere Application Server Version 6.1 client applications must 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 later versions, 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 or later 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 by using the WSDL file from the new service.

An exception occurs because the SDO repository is not configured correctly

If you try to create a Version 6.1 WS-Notification service, and you get the following stack trace, then the 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: smeago1Node03Cell/nodes/smeago1Node03/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:org/CosNaming/NamingContext/NotFound:1.0]] vmcid: 0x0 minor code: 0 completed: No.
```

Multiple messages are 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 usually 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.

Problems can occur when deleting administered subscribers and messaging engines

Deleting administered subscribers

You should be wary of deleting and recreating 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 have 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.

There are differences when using bus destinations with Version 6.1 WS-Notification services

Usually, a bus destination is used as described in Bus destinations. However, this is not the case for Version 6.1 WS-Notification services. The destination that is associated with a Version 6.1 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.

An inbound (application to broker) notification is not successful

Applications that want 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 notification might be unsuccessful because of 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 not succeed include the following:

- Topic is not valid: the topic expression supplied does not match the syntax of the stated dialect, or they specified an unsupported dialect. This is an application error.
- Topic namespace is not valid: the application specifies a topic namespace that has not been configured, but the administrator has specified (on the WS-Notification service) that use of dynamic namespaces is not allowed.
- Topic is not supported: 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”).
- Credentials are not valid: 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 is 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 should monitor this type of exception 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.

An outbound (broker to application) notification is not successful

An outbound web service invocation (broker to application) does not succeed when a remote application is unavailable for invocation. This might be the result of an application failure, a network error, or a firewall configuration issue. When event notifications are not passed to subscribed applications, messages build up on the subscriptions held on the server. The messages held on a given subscription can be observed by 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 does not 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.

How to tidy 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. Usually 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 view the stateful resources (subscriptions and publisher registrations) by 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.

A durable subscription that was created by WS-Notification cannot be deleted when using the service integration bus panel

When you create a subscription by 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 does not succeed. This is 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 by 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 “An inbound (application to broker) notification is not successful” on page 302 and “An outbound (broker to application) notification is not successful” on page 303. 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 (for Version 6.1 WS-Notification services) 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 that use 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 “An inbound (application to broker) notification is not successful” on page 302).
- Subscribe requests that use 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.

You cannot create a Version 6.1 WS-Notification service without a configured SDO repository

When you create a Version 6.1 WS-Notification service, WSDL documents are saved into the SDO repository. You will see the following exception message if you try to create a Version 6.1 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:org.omg/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](#).

An exception occurs when a JAX-WS client that does not have internet access attempts to contact a WS-Notification service

The client application usually resolves the WSDL parts for the service by following web links. If the machine on which the client runs does not have internet access, an exception similar to the following example occurs:

```

WSDLException (at /definitions/import[1]): faultCode=OTHER_ERROR:
Unable to resolve imported document at 'http://docs.oasis-open.org/wsn/brw-2.wsdl',
relative to 'http://localhost:9082/WSNService1WSNServicePt1NB/Service/WEB-INF/wsd1
/NotificationBroker.wsdl': java.net.UnknownHostException: docs.oasis-open.org

```

Configure the client to use a local copy of the WSDL file instead, by following the instructions in the topic [Configuring a JAX-WS client to resolve a WS-Notification service WSDL without following web links](#).

WS-Notification: Known restrictions

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

- “Composition with WS-Policy”
- “Virtual hosts”
- “Optional specification elements” on page 307
- “Interpretation of the specification” on page 307

Composition with WS-Policy

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

Virtual hosts

For WS-Notification applications that are associated with a virtual host, ensure that the virtual host has an alias that uses the host name or an asterisk (*), for example, myHost:9080 or *:9080. The virtual host can have additional separate aliases that use an IP address or the string localhost, but these aliases are not automatically resolved to the host name.

If the virtual host does not have an alias that uses the host name or an asterisk, the following message is produced when the application subscribes to a WS-Notification broker:

```
CWWAR0202E: None of the web services endpoints for this host match the aliases for the virtual host: host_name.
```

This message is written to a log file in the ffcd directory, and to the job log for the servant region.

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”.

Filtering of the following event notifications (selectors) is 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.
- Any filter defined as executed over the message body, except for a filter that uses the XPath 2.0 dialect.

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. However, when registering a demand-based publisher, WebSphere Application Server only supports RegisterPublisher request messages that contain a single topic expression.

Unsupported optional elements

Using the XPath 2.0 dialect to filter event notifications (selectors) is not supported.

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.

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.

Appendix. Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - z/OS

app_server_root

Refers to the top directory for a WebSphere Application Server node.

The node may be of any type—application server, deployment manager, or unmanaged for example. Each node has its own *app_server_root*. Corresponding product variables are *was.install.root* and *WAS_HOME*.

The default varies based on node type. Common defaults are *configuration_root*/AppServer and *configuration_root*/DeploymentManager.

configuration_root

Refers to the mount point for the configuration file system (formerly, the configuration HFS) in WebSphere Application Server for z/OS.

The *configuration_root* contains the various *app_server_root* directories and certain symbolic links associated with them. Each different node type under the *configuration_root* requires its own cataloged procedures under z/OS.

The default is */wasv8config/cell_name/node_name*.

plug-ins_root

Refers to the installation root directory for Web Server Plug-ins.

profile_root

Refers to the home directory for a particular instantiated WebSphere Application Server profile.

Corresponding product variables are *server.root* and *user.install.root*.

In general, this is the same as *app_server_root/profiles/profile_name*. On z/OS, this will always be *app_server_root/profiles/default* because only the profile name "default" is used in WebSphere Application Server for z/OS.

smpe_root

Refers to the root directory for product code installed with SMP/E or IBM Installation Manager.

The corresponding product variable is *smpe.install.root*.

The default is */usr/lpp/zWebSphere/V8R0*.

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.

APACHE INFORMATION. This information may include all or portions of information which IBM obtained under the terms and conditions of the Apache License Version 2.0, January 2004. The information may also consist of voluntary contributions made by many individuals to the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org>. You may obtain a copy of the Apache License at <http://www.apache.org/licenses/LICENSE-2.0>.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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 and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

- access intents
 - Enterprise JavaBeans (EJB)
 - troubleshooting 57
- applications
 - troubleshooting 82

C

- CDI
 - troubleshooting 223

D

- data access
 - troubleshooting 17
- directory
 - installation
 - conventions 231, 309
- dumpNameSpace tool 89
- dynamic cache
 - troubleshooting 45
- dynamic cache service
 - troubleshooting 45

E

- Enterprise JavaBeans (EJB)
 - troubleshooting 53
- errors
 - security configuration 112
 - security enablement 112, 115
 - SSL 125

J

- JNDI
 - dumpNameSpace 86
 - troubleshooting 81
- JPA
 - application logging 62
 - tracing 65
 - troubleshooting 59, 69

- JSP files
 - troubleshooting 220

N

- namespaces
 - dumps 91, 93

O

- ORB
 - troubleshooting 95

S

- SIP
 - tracing 213
 - troubleshooting 211
- SOAP
 - tracing 246
- SPNEGO
 - troubleshooting 143

T

- troubleshooting
 - security authorization providers 132
 - security components 99
 - security configurations 99
 - security enablement
 - access problems 122
 - SPNEGO 143

W

- web applications
 - deployment
 - troubleshooting 219
 - troubleshooting 219
- web services
 - FAQ 247
 - tracing 244
 - troubleshooting 231, 249