WebSphere software

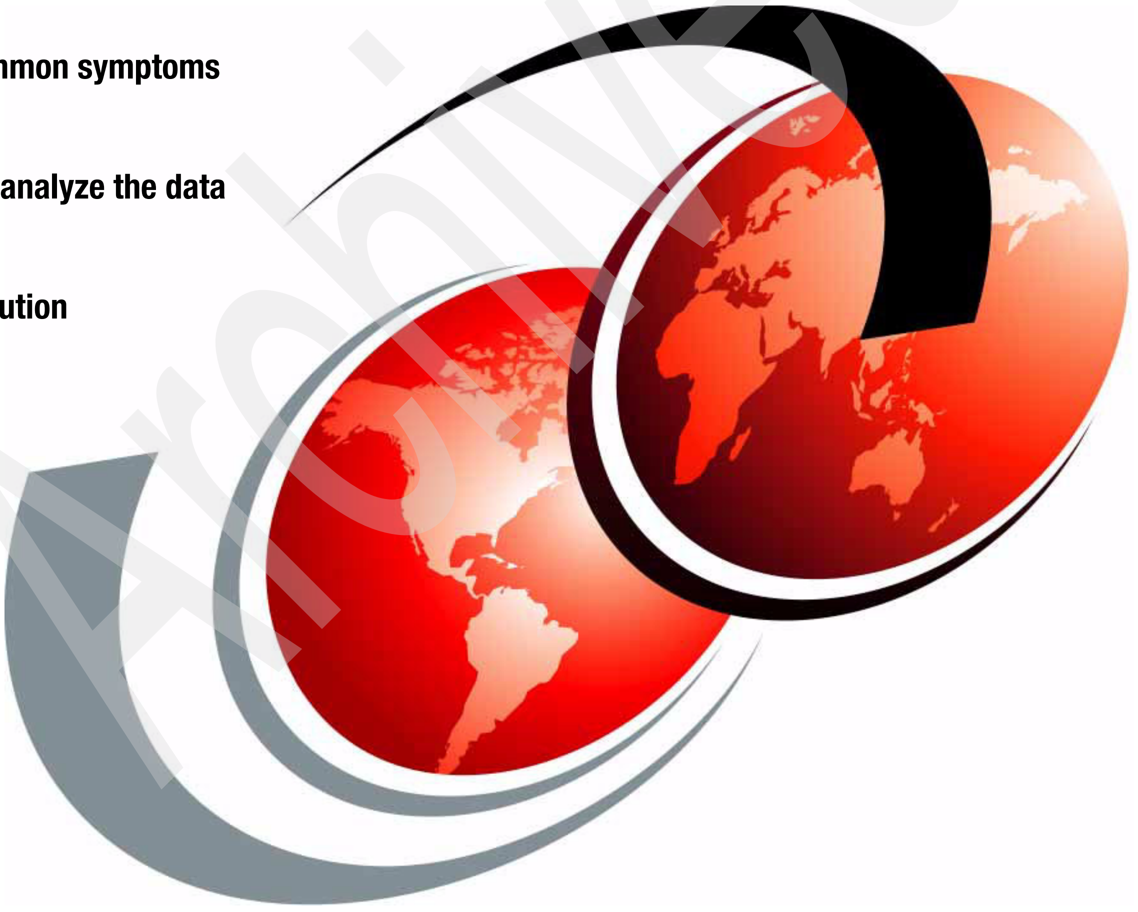
IBM

WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection

Identify common symptoms

Collect and analyze the data

Find a resolution



ibm.com/redbooks

Redbooks

International Technical Support Organization

WebSphere Application Server V6.1 Problem Determination IBM Redpaper Collection

December 2007

SG24-7461-00

First Edition (December 2007)

This edition applies to WebSphere Application Server V6.1.

Contents

Notices

Trademarks

Preface

This IBM® Redbooks® publication is a collection of previously published Redpapers. This publication allows for easy download of all papers.

Each paper addresses the problem determination process to take for specific components on specific platforms. The intent of the papers is to help customers through the process of identifying and resolving problems in WebSphere Application Server V6.1. The reader will be taken through the process of identifying symptoms of the problem, collecting and analyzing data for diagnosing the problem, examining common root causes and solutions for a problem, and finally how to gather documentation before contacting IBM technical support.

As a prerequisite to problem determination efforts on WebSphere® Application Server, you should review the following paper:

- *Approach to Problem Determinations*

  http://www.redbooks.ibm.com/abstracts/redp4073.html

The team that wrote this IBM Redbook

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Carla Sadtler is an IT Specialist at the ITSO, Raleigh Center. She writes extensively about the WebSphere and Patterns for e-business areas. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative. She holds a degree in mathematics from the University of North Carolina at Greensboro.

Gustavo Cezar de Medeiros Paiva is an IT Specialist working on Integrated Technology Delivery SSO, in Hortolandia, Brazil. He has six years of experience in IT. He is an expert in IBM WebSphere Application Server problem determination and certified in WebSphere Application Server V5 and V6. His areas of expertise include WebLogic Application Server, Linux®, AIX®, and Windows® platforms. Gustavo holds a degree in computer science.

John Szuch is a staff software engineer working for System i™ Technical Support in IBM US. He supports Java™, WebSphere Application Server, and

WebSphere Commerce Suite. He has been a member of System i Technical Support since 1997.

Wendy Conti is a Level 3 support software engineer in the United States. She has worked in WebSphere for six years and with the WebSphere Web services engine for over two years. Her area of expertise is in problem determination of the Web services engine.

Jasper Chui is a Support Analyst at the IBM Toronto Software Lab in Toronto, Canada. He has been working with Rational® Application Developer support for two and a half years and specializes in cross product integration issues.

Craig Scott is a Software Support specialist in Australia. He has 18 years of experience in IT and over seven years experience in WebSphere Application Server. For the past three years, he has been working for IBM Software Support Center helping people resolve problems with WebSphere Application Server Version 4 to Version 6.1. He holds a degree in Computer Science from the University of Canberra. His areas of expertise include WebSphere Application Server, WebSphere Edge Server, IBM DB2®, and IBM Content Manager.

Giribabu Paramkusham is a Level 2 Support Engineer in the United States. He has worked in WebSphere for over five years. He is a certified WebSphere Application Server System Administrator. His areas of expertise include Web container problem determination, crash analysis and performance tuning of WebSphere Application Server on AIX, Solaris™, Linux, and Windows platforms.

Robert Larsen is a staff software engineer working for System i Technical Support in IBM US. He supports Java, WebSphere Application Server, and WebSphere Commerce Suite. He is also a subject matter expert for iSeries® for HTTP Powered by Apache, LDAP, and SSL. He has been a member of System i Technical Support since 1999.

Richard Coppen is a Test Architect at the IBM Hursley laboratory, United Kingdom. Since joining IBM in 1997, Richard has been involved with the development of several IBM software products, specializing in the WebSphere product family's Messaging and Web services technologies. Richard received his Masters degree in Software Engineering from the University of Oxford in 2007.

Gareth Bottomley is a Software Engineer in the United Kingdom. He has worked on the WebSphere Application Server family of products for the last six years. He has worked at IBM for nine years. His areas of expertise include J2EE™, JMS data persistence, and transaction processing.

Brian De Pradine is a Developer working for IBM Software Group in Hursley, England. He has worked for IBM for nine years. For the past five years, he has worked in WebSphere Application Server development, specifically in the areas of messaging and Web services support. He has a Bachelor of Science degree in Applied Physics from Columbia University and is currently working towards a Master of Science degree in Software Engineering from Oxford University.

Sarah Drewery is a Software Engineer with WebSphere Enterprise Service Bus (ESB) Foundation Technologies in IBM Hursley, UK. She has eight years experience in testing technologies related to messaging. Her areas of expertise include the WebSphere platform and WebSphere messaging, specifically in Network Deployment and clustering. She has a Bachelor of Engineering degree in Integrated Systems Engineering from Manchester Metropolitan University, and a Master of Science in Bioengineering from the University of Strathclyde.

Graham Hopkins is a Software Engineer in the United Kingdom. He has seven years experience in the messaging field. He has worked for IBM for seven years. His areas of expertise include application servers and messaging.

Philip Nickoll is a Software Engineer in the United Kingdom. He has three years experience in WebSphere and has worked for IBM for ten years. His area of expertise is the JMS resource adapter in WebSphere.

Alasdair Nottingham is a Software Engineer with the IBM WebSphere ESB Foundation Technologies group based in the United Kingdom. He has six years experience in the IT industry working in many fields including J2EE, messaging, security, and SOA, specifically with WebSphere Application Server and WebSphere MQ. He has a Bachelor of Science degree in Computer Science from the University of Southampton (UK).

Matthew Roberts is a Senior Developer in the Service Integration Bus development team, working at the IBM Hursley Lab in the United Kingdom. He has six years experience working for IBM in the enterprise messaging field, initially on the Java Message Service (JMS) component of WebSphere MQ and WebSphere Application Server V5.0, and subsequently on the Service Integration Bus component of WebSphere Application Server V6.0 and above. Matt has technical and team leadership responsibility for a range of Service Integration Bus components including JMS, topology, and routing management, as well as publish/subscribe interoperation with WebSphere Message Broker over the MQLink. Most recently, Matt has been an active member of the OASIS technical committee for WS-Notification, a Web service open standard for publish/subscribe messaging, leading to architectural and development responsibility for implementation of WS-Notification in WebSphere Application Server V6.1.

Dave Vines is an IBM Master Inventor and Software Engineer at the IBM Hursley Laboratory in the United Kingdom working in WebSphere MQ and ESB Development. He joined the laboratory in 1984 and has worked on a wide variety of IBM program products including GDDM®, MQSeries®, LANDP®, and for the past decade, Component Broker and WebSphere Application Server. He received a Bachelor of Science degree from the University of Exeter, England, in 1984.

David Ware is a Senior Software Engineer in the United Kingdom. He has worked for IBM for twelve years. Most of those years, he worked in the field of asynchronous messaging, starting with WebSphere MQ, and for the last five years, on the default messaging provider in WebSphere Application Server. He is an expert in many aspects of messaging and is responsible for the design of the core technologies of the service integration bus messaging engine.

Bryan Williams is a Software Engineer at the IBM Hursley Laboratory in the United Kingdom; he works in WebSphere MQ and ESB Test. He joined the laboratory in 1985 and has worked on a wide variety of IBM program products including SDF, GDDM, MQSeries, IBM Visualization Data Explorer™, WebSphere System Business Integration, and for the past few years, WebSphere Application Server. He has a Bachelor of Science degree in Applied Physics from Coventry University, which he received in 1977. He has worked in various areas of IT for 30 years.

Thanks to the following people for their contributions to this project:

Russell Wright
David Draeger
WebSphere Serviceability Development, IBM Guided Activity Assistant Development, ISA Tools, IBM US

Paul Griffiths
Arun Kumar
Nishant Thakkar
Aaron Walker
Mahesh Rathi
Andrew Lam
Kevin Grigorenko
William A Reichert III
WebSphere Application Server SWAT team, IBM US

Ted Dow
Bob Blom
Jim Shupe
IBM System i software support, IBM US

Isabelle Mauny
IBM Spain

Robert Outlaw
WebSphere z/OS® Level 2 support

David Parsons
WebSphere test team, IBM US

Joseph Mertzlufft
WebSphere Application Server Level 2 support, IBM US

Thanks to the authors of the following book:

- *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798

  Carla Sadtler
  Simon Davitt
  Rana Katikitala
  Thu-Giang Pham
  Craig Scott
  David Titzler
  Hari Venkateshaiah
  Javier Voos

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbooks publication dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

`ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us!

We want our IBM Redbooks publications to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Part 1

Installation problem determination

This part contains the following IBM Redpaper:

- *WebSphere Application Server V6.1 installation problem determination*, REDP-4305 by Carla Sadtler, Gustavo Cezar de Medeiros Paiva, and John Szuch

1

Installation problem determination

Installation issues with WebSphere Application Server can occur at various stages of the process. If you experience any issues during the execution of the launchpad, the installation wizard for WebSphere Application Server, or during a remote or silent installation, you can use this paper to help you diagnose the issue.

The activity that we discuss in this paper walks through the process of debugging the following WebSphere Application Server installation issues:

- Launchpad or installation wizard fails to start
- Installation fails
- Profile creation fails
- IVT fails
- Maintenance update process fails (i5/OS® only)

The activity is applicable to the installation of WebSphere Application Server V6.1 on distributed and i5/OS platforms. The activity does not cover installation of other products or features that shipped with WebSphere Application Server.

1.1 Introduction to the installation process

You can perform the installation of WebSphere Application Server in multiple ways. The methods available depend on the platform on which you are installing the product.

Installation on i5/OS

You can install on i5/OS in one of the following ways:

- ► Install locally on i5/OS (silent installation). This type of installation uses a response file.
- ► Remote installation from a Windows machine using the launchpad and installation wizards. With this type of installation, you enter your responses using a GUI interface.

Installation on distributed systems

You can install on distributed systems in one of the following ways:

- ► Silent installation. This type of installation uses a response file.
- ► Using the launchpad and installation wizard. With this type of installation, you enter responses using a GUI interface.

1.2 Determine problem type

During the installation of WebSphere Application Server, you can run into several potential issues. This activity covers the following issues:

- ► The launchpad or installation wizard fails to start.
- ► The installation process fails.
- ► The profile creation process fails.
- ► Failures occur during the Installation Verification Test (IVT).
- ► Installation of maintenance on i5/OS fails.

1.2.1 Identify the initial symptoms

The first step in diagnosing installation issues is to identify what stage of the installation process is having the problem.

Problems using the launchpad tool or installation wizard

Symptoms of an error using the launchpad or installation wizard include:

- Executing the `launchpad.bat` command in windows or the launchpad.sh script in a UNIX® environment does not start the launchpad.
- Selecting **Launch the installation wizard for WebSphere Application Server** from the launchpad fails to start the installation wizard or the wizard exits with or without an error message.

If you have these symptoms, go to “Launchpad or installation wizard fails to start” on page 6.

Problems with the installation process

Symptoms of an error during the installation process include:

- Installation using the wizard fails.
- During a silent installation, a command prompt returns without errors or messages displayed in the console log.
- INSTCONFFAILED displayed in the installation log.
- Warning or error messages are displayed in the installation wizard or installation log.
- Error or warning messages occur during the installation process that begin with `INST`, `ADMU` (during profile creation), or `WSVR` (during IVT).

If you have these symptoms, go to “Launchpad or installation wizard fails to start” on page 6.

Problems in the profile creation process

You can create a profile during the installation process or later using the profile management tool (distributed platforms) or **`manageprofiles`** command. Indications of an error during profile creation display in the wizard panels and are logged to various logs. Symptoms of a profile creation failure include:

- An error displays in the installation wizard indicating the profile creation failed.
- An error displays when using the profile management tool or when using the **`manageprofiles`** command to create a profile. The error indicates the problem is in the profile creation process.
- You get a `Deployment Manager connection failure` message when creating a custom profile.
- The message `INSTCONFFAILED: The profile could not be created` displays in the profile creation log.

If you have these symptoms, go to “Profile creation fails” on page 20.

Problems in the Installation Verification Tool

You start the Installation Verification Tool (IVT) from the First Steps panel or using the **`ivt`** command after a profile is created successfully. Messages are returned to the wizard panel or the command line and also logged. Symptoms of an IVT failure include:

- `IVTL0075I`: The Installation Verification Tool verification failed.
- `ADMUxxxxE` messages

If you have these symptoms, go to "Installation Verification Test fails" on page 37.

Problems during maintenance on i5/OS

You install WebSphere Application Server maintenance on i5/OS in two stages. First, you apply a group PTF and then run an update script to install the fixpack. You run the update script in batch mode, so the only way to tell if the update is successful is to check the logs that are created during the update. The following entries in the log indicate a failure in the update:

- `INSTCONFPARTIALSUCCESS`
- `INSTCONFFAILED`

If you have these symptoms, go to "Maintenance on i5/OS fails" on page 43.

1.3 Launchpad or installation wizard fails to start

Often, you install WebSphere Application Server on distributed and i5/OS platforms by executing the **`launchpad.bat`** or **`launchpad.sh`** program. On distributed platforms, you execute the launchpad on the system where the installation occurs. When installing on an i5/OS system, you execute the launchpad on a Windows system, and the install occurs remotely to the i5/OS system. When started, the launchpad presents a GUI interface that allows you to select installation activities, including the option to start the installation wizard for WebSphere Application Server.

1.3.1 Steps to diagnose the problem

The following actions can help you to diagnose a problem with the launchpad or installation wizard:

- Verify that the prerequisites for the installation are in place.
- Verify that you have a supported Web browser and Web browser configuration.

- If the problem is with the launchpad, attempt to bypass the launchpad by executing the installation wizard directly.
- If the problem is with the installation wizard, attempt to bypass the installation wizard by doing a silent install.

These steps are explained more fully in the sections that follow. If after you have read these sections the information here does not resolve your problem, collect and analyze the log files.

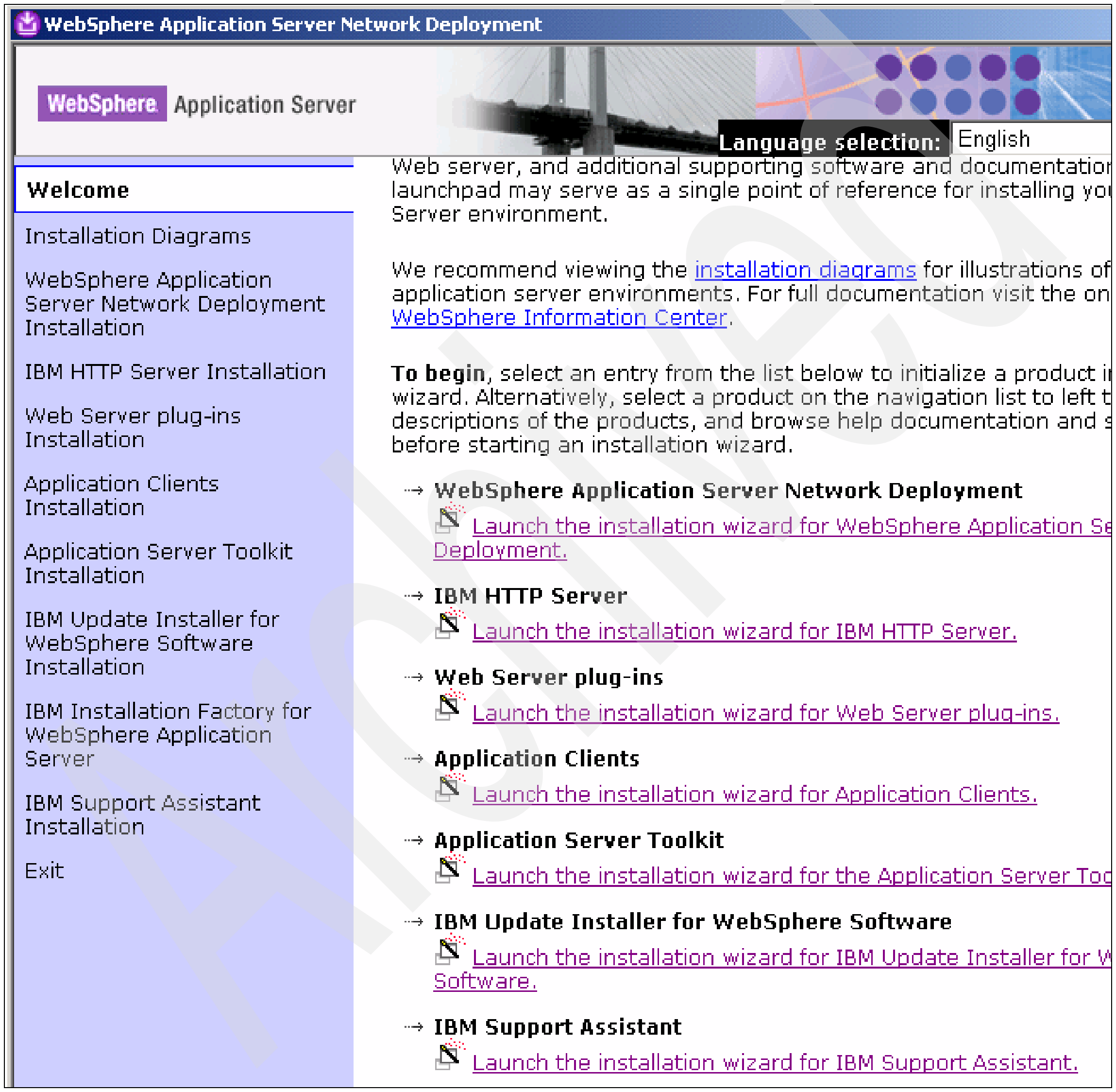
1.3.2 Check system integrity

The launchpad executable is located in the root directory of the WebSphere Application Server installation CD. Executing it should bring up a user interface that provides the option to start various installation activities, including the WebSphere Application Server installation wizard.

Possible causes for problems with the launchpad or installation wizard include Web browser requirements and disk space or permission requirements.

Attempt to start the launchpad

Start the launchpad. If the launchpad starts correctly, you should see a window that looks similar to Figure 1-1. The options available will depend on the WebSphere Application Server package. If the launchpad fails to start, note any error messages that display.



WebSphere Application Server Network Deployment

WebSphere Application Server

Language selection: English

Welcome
Installation Diagrams
WebSphere Application Server Network Deployment Installation
IBM HTTP Server Installation
Web Server plug-ins Installation
Application Clients Installation
Application Server Toolkit Installation
IBM Update Installer for WebSphere Software Installation
IBM Installation Factory for WebSphere Application Server
IBM Support Assistant Installation
Exit

Web server, and additional supporting software and documentatio
launchpad may serve as a single point of reference for installing you
Server environment.

We recommend viewing the installation diagrams for illustrations of
application server environments. For full documentation visit the on
WebSphere Information Center.

To begin, select an entry from the list below to initialize a product i
wizard. Alternatively, select a product on the navigation list to left t
descriptions of the products, and browse help documentation and s
before starting an installation wizard.

→ **WebSphere Application Server Network Deployment**
Launch the installation wizard for WebSphere Application Se
Deployment.

→ **IBM HTTP Server**
Launch the installation wizard for IBM HTTP Server.

→ **Web Server plug-ins**
Launch the installation wizard for Web Server plug-ins.

→ **Application Clients**
Launch the installation wizard for Application Clients.

→ **Application Server Toolkit**
Launch the installation wizard for the Application Server Too

→ **IBM Update Installer for WebSphere Software**
Launch the installation wizard for IBM Update Installer for W
Software.

→ **IBM Support Assistant**
Launch the installation wizard for IBM Support Assistant.

Figure 1-1 Launchpad for WebSphere Application Server Network Deployment

Verify the prerequisites for installation

Problems starting the launchpad or installation wizard can usually be traced back to missing prerequisite system or application levels. Verify that you have the proper prerequisites for installation. You can find this information at:

`http://www-306.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

Attempt to start the installation wizard

The launchpad simply gives you a GUI interface to the installation activities. Verify that you can start the installation wizard:

1. Start the launchpad.
2. If the launchpad starts correctly, select the link to launch the installation wizard for WebSphere Application Server.
3. If the launchpad does not start, start the installation wizard directly by executing install.exe (Windows) or install (UNIX) in the WAS directory on the installation CD.

Attempt a silent installation

If the installation wizard will not start, you can try the following alternatives:

- On distributed systems, you can use the silent installation method. For information on the silent installation method, see *Installing silently*, which is available at:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_runSilent.html`

- On i5/OS, you can use a remote silent installation. For information on this type of installation, see Installing from a Windows workstation command line at:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.iseries.doc/info/iseriesnd/ae/tins_is_instrmt.html`

If the installation wizard does not start or you want to continue with problem determination for the launchpad, continue with this activity.

Verify your Web browser type, level, and configuration

If you are having problems starting the launchpad, it is possible that you do not have a supported Web browser installed or configured properly. Steps to ensure you have the proper Web browser environment are:

- Verify the browser type and level are supported
- Export the browser location (UNIX platforms)
- Enable JavaScript

Verify the browser type and level are supported

Mozilla and Internet Explorer® are supported for configuration and installation activities. You can find a current list of supported browsers, their supported levels, and installation information for each platform in the Information Center article, *Using the launchpad to start the installation* , Network Deployment on distributed systems, which is available at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/tins_launch.html`

This URL is for Network Deployment on distributed platforms. To find comparable topics for other packages and platforms, search the Information Center using `tins_launch` as the search argument.

Export the browser location (UNIX platforms)

On UNIX platforms, ensure that the location of the supported browser is exported. For example, if the Mozilla executable is located in the /usr/bin directory, then export its location as follows:

```
export BROWSER=/usr/bin/mozilla
```

Enable JavaScript

Ensure that JavaScript™ is enabled in the browser options or preferences. For example:

- In Mozilla for AIX, select **Edit** → **Preferences** → **Advanced** → **Scripts & Plugins** - Enable JavaScript for: Navigator, Allow Scripts to: Select all options.
- In Internet Explorer, select **Tools** → **Internet Options** → **Security** → **Internet** → **Custom Level** → **Scripting** → **Active scripting**. Select **Enable.**
- In Mozilla Firefox, select **Tools** → **Options** → **Content**. Select **Enable JavaScript** and click **Advanced**. Select all options.

1.3.3 Collect diagnostics

Collect the installation log as described in “Installation log” on page 45.

1.3.4 Analyze diagnostics

The installation log includes messages with details of the installation process and can contain messages that indicate a problem.

Examine the installation log

Look for entries that include `INSTCONFSUCCESS`, `INSTCONFPARTIALSUCCESS`, or `INSTCONFFAILED`, which might indicate the current status of the installation. Make note of the log entries that indicate partial success or failure and any entries preceding them that might indicate the problem.

Evaluate the results

The installation log includes messages that indicate steps failed or were only partially successful. If you see the `INSTCONFPARTIALSUCCESS` or `INSTCONFFAILED` messages, you should also see error or warning messages preceding them (for example, messages that indicate problems with resources, such as not enough disk space, exceptions in the JVM™, segmentation faults, and so forth.

If you received messages that are self-explanatory, attempt to correct the problem and retry the launchpad or installation wizard.

If you received messages that are not self explanatory, go to “The next step” on page 48 to perform online support searches using these messages as the basis of your search.

Suitable JVM could not be found

An error that resembles the following error indicates a problem with disk space:

```
A suitable JVM could not be found. Please run the program again using
the option -is:javahome < JAVA HOME DIR> No space left on device
```

This error indicates that there is not enough free space for the installer to run on. You can get this error even if enough space exists where you plan to install WebSphere Application Server (for example, drive D: or /usr).

Verify that the location of C:\Temp or %TEMP% in Windows or /tmp directory in UNIX has enough free space for the installer to run. You need to check the installation document to determine the exact amount of temporary disk space required, typically a minimum of 100 MB.

Another option is to use **`-is:`*`tempdir`*** with the installation wizard, where *tempdir* is the location of a temporary directory on a partition with enough free space.

1.3.5 Validate the solution

To verify that the problem is resolved, restart the launchpad. If it starts correctly, select **Launch the installation wizard for WebSphere Application Server** from the launchpad. If you still have problems, go to “The next step” on page 48.

1.4 Installation fails

Installation errors can have different underlying causes. For example:

- Insufficient file permissions for the user ID doing the installation
- Insufficient disk space for the product or install process
- Missing prerequisites
- Incorrect response file configuration during a silent install
- Errors due to limitations in non-root installations

This activity helps you to identify the root cause of your installation error and how to resolve it.

Non-root installation

You can now install WebSphere Application Server V6.1 on distributed systems using a non-root user ID. You can use this feature with both silent and interactive mode for full product installations and removals. This type of installation has the following limitations that can be interpreted as errors:

- Product cannot be registered using the native operating system mechanisms
- Port conflicts cannot be detected against other users' installations
- Ports must use values greater than 1024
- Uninstallation and update installation can only be performed by original user, group member or root
- Windows services will not be created during installation

1.4.1 Identify symptoms

When using the installation wizard, some errors are caught and displayed before the installation begins. Messages that you would normally see in the installation wizard are logged to the installation log when you are doing a silent installation.

Errors messages during the wizard

The following messages are some of the messages that you might see that indicate missing prerequisites or invalid system conditions:

- `Could not establish a connection to the i5/OS server`
- `Directory could not be validated as a writable directory`
- `There is insufficient free disk space on the system`
- `Current user profile needs authorities: *ALLOBJ *SECADM`
- `Your operating system failed the prerequisites check`

If you see one of these messages, see “Missing prerequisites or invalid conditions” on page 16 for more information.

If you do not have a symptom displayed, are using a silent installation, or have a symptom that we do not list here, you need to collect diagnostics to continue.

1.4.2 Collect diagnostics

To collect diagnostics, collect the installation log as described in “Installation log” on page 45.

1.4.3 Analyze the diagnostics

In the installation log, look for messages that include the following lines:

- `INSTCONFSUCCESS`
- `INSTCONFFAILED`
- `INSTCONFPARTIALSUCCESS`

If you find any messages with `INSTCONFFAILED` or `INSTCONFPARTIALSUCCESS`, scan the logs to see what was taking place and to see whether you can identify any messages that indicate the problem. In particular look for the following text:

- `com.ibm.ws.install.ni.ismp.actions.WasSilentInstallInputValidationAction, err`
- `com.ibm.ws.install.ni.ismp.actions.MaintenancePrereqCheckAction, err`
- `com.ibm.ws.install.ni.ismp.actions.ISMPWarningDialogAction, wrn`
- `CWUPIxxxxE` messages

The stack trace messages indicate the error.

Evaluate the messages

When you evaluate the message, you can look for the following issues:

- The installation was successful but the profile creation failed
- Silent installation or local installation on i5/OS response file problems
- Error or warning messages indicating invalid prerequisite or conditions

The installation was successful but the profile creation failed

If you chose to create a profile during the installation process, it is possible for the installation to complete successfully but for the profile creation to fail as illustrated in Example 1-1.

Example 1-1 Installation successful, profile creation fails

```
(Mar 14, 2007 11:46:51 AM), Process,
com.ibm.ws.install.ni.ismp.installtoolkitbridge.ISMPInstallToolkitBridg
eForNIFramework, wrn, Config action failed: 97SInstallInvokeWSProfile -
e:\WebSphere\AppServer\properties\version\nif\config\install\97SInstall
InvokeWSProfile.ijc
(Mar 14, 2007 11:46:52 AM), Process,
com.installshield.wizard.platform.win32.Win32PPKRegistryServiceImpl,
dbg.registry, writing VPD to E:\WINNT\vpd.properties
(Mar 14, 2007 11:46:58 AM), Process,
com.ibm.ws.install.ni.ismp.actions.SettleNIFRegistryAction, msg1,
Current install/uninstall process is successful. Process type is:
install
(Mar 14, 2007 11:46:58 AM), Process,
com.ibm.ws.install.ni.ismp.actions.SetExitCodeAction, msg1, CWUPI0000I:
EXITCODE=2
(Mar 14, 2007 11:46:58 AM), Process,
com.ibm.ws.install.ni.ismp.actions.ISMPLogSuccessMessageAction, msg1,
INSTCONFPARTIALSUCCESS
```

If you have this type of error, it is important to note that the installation was successful and that you do not need to repeat it. You can create the profile independently using the profile management tool or the **manageprofiles** command.

Where to go from here: If your installation was successful but the profile creation failed, go to "Profile creation fails" on page 20.

Silent installation or local installation on i5/OS response file problems

If you see any of the following messages, go to "Response file problems" on page 18:

- `LICENSE_NOT_ACCEPTED: Accept the license agreement in the response file before installing.`
- `A non-valid installation directory was specified.`
- `adminUserName, adminPassword not specified.`
- `NONROOT_INSTALL_DISALLOWED : Set the non-root install allowed setting to true in the response file before installing.`

Error or warning messages indicating invalid prerequisite or conditions

The following messages are some that indicate a missing prerequisite or invalid condition for the installation:

- `Could not establish a connection to the iSeries server`
- `Directory could not be validated as a writable directory`
- `There is insufficient free disk space on the system`
- `Current user profile needs authorities: *ALLOBJ *SECADM`
- `Your operating system failed the prerequisites check`

If you see one of these messages, see "Missing prerequisites or invalid conditions" on page 16.

Review error message user response suggestions

Review the text and user response information for these messages for possible solutions. You can find the message text and user response information at CWUPI messages, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.install.ni.framework.resourcebundle.NIFResourceBundle.html

Where to go from here: If you found messages that we do not list here, go to "The next step" on page 48.

1.4.4 Missing prerequisites or invalid conditions

This section discusses missing prerequisite conditions that can cause a problem during installation.

Could not establish a connection to the i5/OS server

This symptom indicates a problem accessing the iSeries server from the Windows system when doing a remote install to iSeries. The following issues can cause the remote connection from the installation wizard to the i5/OS system to fail:

- The host servers are not started on the iSeries system.

  To correct this issue, run the **`STRHOSTSVR *ALL`** command.
- The host name is not specified or configured in TCPCFG.

  To correct this issue, verify that the iSeries host name is listed in `CFPTCP option 10`.
- Remote connection fails when the IP address is used.

  To correct this issue, try using the host name instead.

Directory could not be validated as a writable directory

This error usually indicates a file permissions error. You usually see this error in non-root installations on UNIX systems. To correct this issue, ensure that the user ID that performs the installation has write permission for the installation location and for the location where the profile is created.

Relevant topics in the Information Center that are related to this issue are:

- *Creating profiles for non-root users*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tpro_manage_nonroot.html
- Preparing Linux systems for installation

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_linuxsetup.html

Correct the permissions and go to "Validate the solution" on page 12.

There is insufficient free disk space on the system

This message indicates there is not enough free disk space on the system for the installation. A variation of this message indicates there is not enough free space for the profile to be created:

```
There is insufficient free disk space on the system for profile
creation.
```

This message gives you the information that you need. It lists the location where the space is needed and how much is needed. To correct this issue, you need to free enough disk space for the profile or specify a different location and attempt to create the profile using the profile management tool or the **`manageprofiles`** command.

Current user profile needs authorities: *ALLOBJ *SECADM

If you have this issue, you need to ensure that the following attributes are true for the user ID that is used to install the product:

- The user ID has ALLOBJ authority.
- The user ID is enabled.
- The password has not expired.

Correct these conditions and go to "Validate the solution" on page 12.

Your operating system failed the prerequisites check

The installation process performs a prerequisite check before attempting the installation. If your system does not have the proper prerequisite programs or conditions (for example space), the check will fail. The sections that follow list some of the common problems and where to go to find a current list of the required prerequisites.

i5/OS

Some of the more common reasons that the prerequisite check can fail on i5/OS include:

- JDK™ 1.5 is not installed
- The group PTF for Java not applied
  - For V5R3: SF99269
  - For V5R4: SF99291

  You can download these products from Fixcentral at:

  `http://www-912.ibm.com/eserver/support/fixes/fixcentral/main/iseries/`

  You can find a list of fixes on the installation CD:

  cd_root/WAS/lib/V5R3PTFs and *cd_root*/WAS/lib/V5R4PTFs

You can find a complete list of i5/OS prerequisites in the topic *iSeries prerequisites*, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.iseries.doc/info/iseries/ae/cins_is_prqsvr.html

Distributed

For information about the current prerequisites, see *WebSphere Application Server detailed system requirements*, which is available at:

http://www-306.ibm.com/software/webservers/appserv/doc/latest/prereq.html

For information about preparing your system for installation, see *Preparing the operating system for product installation*, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_prepare.html

This URL is for the Network Deployment package. If you are installing a different package, search the Information Center for the page name, *tins_prepare.html*, and select the topic that is appropriate for your package.

1.4.5 Response file problems

Problems that occur during a silent installation are often a result of incorrectly specified options in the response file. This section covers the following:

- License agreement was not accepted or found
- Non-root installation disallowed
- Non-valid installation directory
- (iSeries) adminUserName, adminPassword not specified

You can find information about installing on i5/OS using a response file in *Installing WebSphere Application Server from your iSeries server*, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.iseries.doc/info/iseries/ae/tins_is_instloc.html

You can find information about customizing the response file for distributed systems in responsefile.nd.txt, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rins_responsefile_nd61.html

The following sections describe some of the problems that you might see.

License agreement was not accepted or found

There are several possible causes for this error, which include:

- Ensure the line `-OPT silentInstallLicenseAcceptance` in the response file is set to `true`.
- File not found error.

  Ensure that the location of the modified response file matches the `-options path/responsefile` parameter in the installation script.
- (i5/OS) Cannot find the responsefile on the media.

  The CD media and DVD media have different paths for the responsefile. For CD media, the response file is in /QOPT/WEBSPHERE/WAS/responsefile.base.txt. For DVD media, the response file is in /QOPT/WEBSPHERE/os400_ppc64/WAS/responsefile.base.txt.

Non-root installation disallowed

If you receive this error, set the `-OPT allowNonRootSilentInstall` parameter in the response file to `true` for non-root silent installation.

Non-valid installation directory

If you receive this error, change the `-OPT installLocation` parameter in the response file and set this with a correct directory location.

(iSeries) adminUserName, adminPassword not specified

By default, the `PROF_enableAdminSecurity` option in the responsefile is set to `true`. If you want to enable administrative security for the default profile that is created during the installation, you must specify values for the `PROF_adminUserName` and `PROF_adminPassword` options. If you do *not* want to enable administrative security for the default profile, change the value for the `PROF_enableAdminSecurity` option from `true` to `false`.

You specify these options in the INSTALL command. For example, from QSHELL, the command might look like this:

```
INSTALL -options path/responsefile -PROF_adminUserName myuser
-PROF_adminPassword mypassword
```

The user ID and password do not need to be a system user ID and password or an LDAP user ID and password. The ID-and-password pair that you specify are stored in the user registry and are used for administrative security for the default profile.

1.4.6 Validate the solution

To validate the solution, attempt to install the product. If the installation detects that the product is already installed, you need to uninstall it first. If the installation completed but profile the creation failed, this step is not necessary. The installation is complete, and you can use the profile management tool or the **`manageprofiles`** command to attempt to recreate the profile. For help on diagnosing profile creation problems, go to “Profile creation fails” on page 20.

Uninstalling the product: For information about how to uninstall the product, consult the following Web site:

- For i5/OS

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.iseries.doc/info/iseries/ae/tins_is_uninstall.html

- Distributed platforms

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_uninstall.html

These topics apply to Network Deployment package. If you are installing another package, you can search the Information Center using *tins_uninstall* as the search argument and then select the appropriate topic for your package.

1.5 Profile creation fails

Profiles allow you to define multiple runtime environments, each with its own administrative interface, while sharing the same code base. Problems with profile creation can be indicated with messages that display when running the profile management tool. These problems might be due to long directory paths, file permissions, problems with the host name, and so forth. A WebSphere Application Server installation is not functional until at least one profile is created.

The symptoms that you encounter during profile creation can vary depending upon how you create the profile:

- After installation using the profile management tool
- After installation using the **`manageprofiles`** command
- As part of WebSphere Application Server installation (which uses **`manageprofiles`**)

This activity covers problems that occur during the profile creation process and federation of custom nodes. If your problem occurs after profile creation, for example, the new application server or deployment manager will not start, see "Installation Verification Test fails" on page 37.

Creating profiles

If you are installing the base WebSphere Application Server or Express package, an application server profile is created automatically. If you are installing the Network Deployment package, you are given the option to create a profile.

Profiles are created using the `manageprofiles` command. During installation, this command is invoked without your knowledge. To create a profile after installation, you can use the GUI interface to this command (profile management tool), or you can invoke the command directly.

Profile types

The types of profiles that are available to you depend upon the WebSphere Application Server package that you have installed:

- **Application server profile**: An application server profile defines one application server called *server1*. The default applications are installed.
- **Deployment manager profile:** The deployment manager profile creates the deployment manager process (`dmgr`). The deployment manager provides centralized administration of multiple application server nodes and custom nodes as a single cell. The deployment manager provides administration for basic clustering and caching support, including failover support and workload balancing.
- **Custom profile:** A custom profile defines an empty node that you must federate to a cell. Federation is done using the `addNode` command. When federated, you can use the administration tools to customize the node by creating servers and clusters. The node does not include a default application server or default applications.
- **Cell profile:** You can use a cell profile to create a deployment manager, a federated node, and an application server on that node on a single system. It creates two profiles, one for the deployment manager and one for the node and application server.

For Base and Express installations, the application server profile is the only profile allowed.

To learn more about profiles

To learn more about profiles and how to create or manage them, the following resources might be useful:

- *WebSphere Application Server V6.1: System Management and Configuration, SG24-7304*

  `http://www.redbooks.ibm.com/abstracts/sg247304.html`
- *WebSphere Application Server V6.1: Planning and Design, SG24-7305*

  `http://www.redbooks.ibm.com/abstracts/sg247305.html`
- *Creating and deleting profiles*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tpro_profiles.html`

1.5.1 Identify symptoms

If you are using the installation wizard, the profile management tool, or the **`manageprofiles`** command, error indications are often displayed directly to you. If you are using a wizard, input is validated and error messages display in the window where an incorrect parameter is entered. Other messages are displayed in the Results panel. In the case of the **`manageprofiles`** command, error messages display in the command window. It is important that you make a note of any of these messages for use in diagnosing the problem.

Typical examples of the types of error messages that you might see include:

- `Profile creation failed.`
- Custom profile errors when connecting to the deployment manager:
  - `Unable to connect to the deployment manager using the host name and port`
  - `Unable to connect to the deployment manager using the host name and port using the specified user name and password.`
  - `Authentication failed for deployment manager connection.`

  If you see any of these messages, go directly to "Custom profile federation error" on page 31.
- `Activity was detected on these ports.` (And the port listed is numbered less than `1024`.)

  If you have this message, go directly to "Invalid ports selected" on page 35.

1.5.2 Collect diagnostics

The next step in the diagnostic process is to collect the logs that are produced during installation and profile creation.

The following logs can help you to determine whether the profile was created, and if it was not, at what stage the process failed:

- Profile management tool log

  This log is only required if you are using the profile management tool but have not captured the error messages displayed. For information about collecting this log, see "Profile management tool log" on page 47
- Profile creation log

  For instructions on collecting this log, see "Profile creation log" on page 46
- Installation log

  For information about collecting this log, see "Installation log" on page 45
- addNode.log

  This log is only required if you are creating a custom profiles and have selected the option to federate the node to the cell. For information about collecting this log, see "addNode command log" on page 47.
- (i5/OS) SystemOut JVM log

  This log is only required if you are installing to an i5/OS system. For information about collecting this log, see "JVM logs" on page 47.

1.5.3 Analyze diagnostics

The next step is to examine each log for messages that indicate the state of the profile creation and possible errors.

Examine the profile management tool log

This log includes messages that are generated during the profile creation. Look for records with the following text and note the error type:

- `<level>WARNING</level>`
- `<level>SEVERE</level>`
- `<message>Setting ERROR message =`
- `CPWKI0314E`
- `ADMCxxxxE` messages

Validation errors

The profile management tool validates the input and logs any errors both to the wizard console and to the profile management tool log.

Example 1-2 shows the error that is generated when a profile name is specified that already exists or when the directory for the profile exists.

Example 1-2 Validation error in the profile management tool log

```
<record>
  <date>2007-03-08T15:21:57</date>
  <millis>1173385317140</millis>
  <sequence>1151</sequence>
  <logger>com.ibm.ws.profile.validators.DirectoryValidator</logger>
  <level>SEVERE</level>
  <class>com.ibm.ws.profile.validators.DirectoryValidator</class>
  <method>runValidator</method>
  <thread>10</thread>
  <message>The profile path is not valid.</message>
</record>
```

When you see errors that indicate a problem with the parameters that are specified, repeat the profile creation using the profile management tool or the **`manageprofiles`** command and correct the input.

If you see the following message, go directly to "Invalid ports selected" on page 35:

```
Activity was detected on these ports. (And the port listed is
numbered less than 1024.)
```

Errors during the federation of a custom profile

Errors indicating problems during the federation of a custom profile are logged in both the profile management tool log and in the addNode log. Examples of federation errors include:

- `Unable to connect to the deployment manager using the host name and port`
- `Unable to connect to the deployment manager using the host name and port using the specified user name and password.`
- `Authentication failed for deployment manager connection.`

Example 1-3 indicates a problem with the connection to the deployment manager.

Example 1-3 Federation error in the profile management tool log

```
<message>null**CWPKI0314E:** The following error is returned from an
exception: **ADMC0016E: The
          system cannot create a SOAP connector to connect to host
          192.168.1.101 at port 8881.**
```

If you see this type of error, go to "Custom profile federation error" on page 31.

Review error message user response suggestions

Review the text and user response information for these messages for possible solutions. You can find the message text and user response information at ADMC messages, which is available online at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.management.resources.connector.html

Examine the profile creation log

Events that occur during profile creation are recorded to the profile creation log. This log file is created when the installation phase has completed the file copy process and starts creating a default profile. This log file is also created whenever the profile management tool or the **`manageprofiles`** command is executed.

This log file is an XML log file and is best viewed by a viewer that can format XML, for example a Web browser or WordPad in Windows. The entries in this log file consist of `<record>` entries.

What to look for in this log file

Search the log file for the following entries:

- `<level>WARNING</level>`
- `<level>SEVERE</level>`
- `<message>INSTCONFSUCCESS: Success: Profile` *`profile_name`* `now exists`
- `<message> INSTCONFFAILED`
- `<message> INSTCONFPARTIALSUCCESS`
- `<message>`*`text`*` - FAILURE</message>`

If you find `INSTCONFFAILED` or `INSTCONFPARTIALSUCCESS` messages, look at the messages that precede these messages to see what the process was doing when it failed. Warning messages can occur even in a successful profile creation. These warning messages are only interesting if you do not found any other indication of the error.

A sample log entry that indicates an error would look similar to that shown in Example 1-4.

Example 1-4 Example of an error entry

```
<record>
  <date>2007-03-08T11:00:35</date>
  <millis>1173369635312</millis>
  <sequence>14855</sequence>

<logger>com.ibm.ws.profile.cli.WSProfileCLICreateProfileInvoker</logger
>
  <level>SEVERE</level>

<class>com.ibm.ws.profile.cli.WSProfileCLICreateProfileInvoker</class>
  <method>executeWSProfileAccordingToMode</method>
  <thread>10</thread>
  <message>INSTCONFFAILED: The profile could not be created.  For more
information, consult the
C:\WebSphere\AppServer\logs\manageprofiles\create.log file.</message>
</record>
```

Example of a successful completion

Example 1-5 is an example of a log entry indicating a successful profile creation:

Example 1-5 Log record of successful profile creation

```
<record>
  <date>2007-02-26T14:36:47</date>
  <millis>1172518607687</millis>
  <sequence>5651</sequence>
<logger>com.ibm.ws.profile.cli.WSProfileCLICreateProfileInvoker</logger
>
  <level>INFO</level>
<class>com.ibm.ws.profile.cli.WSProfileCLICreateProfileInvoker</class>
  <method>executeWSProfileAccordingToMode</method>
  <thread>10</thread>
  <message>INSTCONFSUCCESS: Success: Profile Custom01 now exists.
Please consult
C:\WebSphere\AppServer\profiles\Custom01/logs/AboutThisProfile.txt for
more information about this profile.</message>
</record>
```

Evaluate the profile creation log results

Using the messages you noted from the logs, evaluate the symptoms.

Profile creation was successful

If you found the message `INSTCONFSUCCESS: Profile profile_name now exists` in the log file, this message indicates that the profile creation was successful and that you probably do not have a profile creation problem.

If your problem occurs after profile creation, during the installation verification test or when you attempt to start the new server, go to "Installation Verification Test fails" on page 37.

File path length error

If you are installing on a Windows platform, the following issues can indicate a path length problem:

- You do not have a *profile_name*_create.log, but rather, you have create.log.
- You see any of the following messages in the create.log:
  - `<message>Task stopped for: mkdir - FAILURE</message>`
  - `<message>Target stopped for: copyTemplate - FAILURE</message>`

If you see this error, go to "File path length error" on page 30.

Template path error

If you are installing in a UNIX environment, the error illustrated in Example 1-6 can indicate a permissions error or incorrect specification of the template path directory. In Example 1-6, the template name was specified incorrectly. The template name is *default*, but *defaul* was entered in the command.

Example 1-6 Template path error - improper template name

```
<record>
  <date>2007-03-13T11:26:33</date>
  <millis>1173795993062</millis>
  <sequence>38</sequence>
  <logger>com.ibm.wsspi.profile.WSProfileException</logger>
  <level>WARNING</level>
  <class>com.ibm.wsspi.profile.WSProfileException</class>
  <method>WSProfileException</method>
  <thread>10</thread>
  <message>Exception message is: Cannot locate the template: No profile
template exists at path
E:\WebSphere\AppServer\profileTemplates\defaul.</message>
</record>
```

In Example 1-7, the `-templatePath` parameter was entered incorrectly. The user entered `-templatesPath`, when the proper parameter is `-templatePath`.

Example 1-7 Template path error - improper parameter specified

```
<message>Incoming command line is: { "-create" ,"-help" ,"-templatesPath"
,"/opt/WebSphere/AppServer/crso/profileTemplates/managed" }</message>
<message>Could not resolve templatePath from command line</message>
```

This error also causes the following response in the **`manageprofiles`** window:

```
Cannot locate the template: No profile template exists at path...
```

If you see this message, go to "Template path error" on page 30.

Examine the installation log

The installation log includes error messages that are generated during installation, for example, those errors that are detected through the prerequisite check. In addition to indicating whether the installation was successful and whether the profile creation was successful, you might see messages that indicate that prerequisites or system conditions have not been met.

Specifically, you should look for `CWUPIxxxxE` messages.

Insufficient free disk space

The following message indicates a problem with free disk space:

```
CWUPI0033E: There is insufficient free disk space on the system for
profile creation.
```

This message gives you the information that you need. It lists the location where the space is needed and how much is needed.

To correct this issue, you need to free enough disk space for the profile or specify a different location and attempt to create the profile using the profile management tool or the **`manageprofiles`** command.

Review error message user response suggestions

Review the text and user response information for these messages for possible solutions. You can find the message text and user response information at CWUPI messages, which is available at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.install.ni.framework.resourcebundle.NIFResourceBundle.html`

Examine the addNode log

Look for the following message types:

- `ADMCxxxxE`
- `ADMUxxxxE`
- `ADFSxxxxE`

Evaluate the results

Any of the following messages indicate a problem connecting to the **`dmgr`** process. This connecting to the **`dmgr`** process is required when the custom profile is federated to the deployment manager.

- `ADMU0006E: Exception creating Deployment Manager connection: com.ibm.websphere.management.exception.ConnectorException:`
- `ADMC0016E: The system cannot create a SOAP connector to connect to host` *`host_name`* `at port` *`port_number`*`.`
- `ADFS0112E: File transfer has failed with the following message: Upload retry limit exceeded for file` *`file_name`* `Exception: java.net.UnknownHostException:` *`host_name`*`.`
- `ADMU0011E: Error creating configuration in the cell repository com.ibm.websphere.management.filetransfer.client.TransferFailedException: Upload retry limit exceeded for file` *`file_name`* `Exception: java.net.UnknownHostException:` *`host_name`*
- `ADMN0022E: Access is denied for the getTokenForNodeFederationOrRemoval operation on AdminOperations MBean because of insufficient or empty credentials.`

If you see one or more of these errors, go to "Custom profile federation error" on page 31.

Review error message user response suggestions

Review the text and user response information for these messages for possible solutions. You can find the message text and user response information at:

- ADMC messages

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.management.resources.connector.html`

- ADMU messages

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.management.resources.nodeutils.html`

- ADFS messages

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.management.resources.fileservice.html

Examine SystemOut for a TCP configuration error

One particular problem that can occur on i5/OS is due to a TCP configuration error. Scan the systemOut log for the following error message:

```
Unrecognized host name
```

This message indicates that TCP is not configured properly. Option 12 in CFGTCP must match (including case and being fully qualified) option 10 in CFGTCP.

1.5.4 Template path error

This error is a result of an incorrect path specified for the `manageprofiles -templatePath` parameter. Because the user does not specify this parameter during the installation process, this error can only occur when you are using the **`manageprofiles`** command.

Four templates are located in *app_server_root*/profileTemplates directory. These templates are cell, default, dmgr, managed. The profile management tool uses these templates as models to create profiles.

If you receive this error, verify that the `-templatePath` parameter has been specified correctly and that the user has permission to access the directory.

1.5.5 File path length error

When long path names are used on Windows operating systems as the installation path for WebSphere Application Server or for the profile location, the overall length of the commands or files that are created can exceed the operating system length.

An error is generated if you enter either:

- An installation directory path that is longer than 60 characters
- A profile directory path and profile name that together are longer than 80 characters

It is possible that even if you specify names that meet this criteria, you can run into the problem during profile creation as the components of the profile are created.

Resolve the problem

To resolve this problem, you have the following options:

- Reinstall the product using a shorter path for the installation directory.
- Create the profile after installation using a shorter path for the profile location.

  The default path using during installation for profiles is *app_server_root*/profiles. If you run into this problem while creating a profile during installation, select `None` as the choice for creating the profile and create it after installation using the profile management tool or the **`manageprofiles`** command. Both options allow you to choose a profile location.
- Choose short names for the cell, node, and server when you create the profiles. The default names, for example dmgr01, are usually short enough.
- Edit the *app_server_root*\bin\setupCmdLine.bat file to make it use the Windows **`subst`** command. The **`subst`** command maps an entire path to a virtual drive. After editing the file, run the profile management tool again.

Tip: It is always a good idea to use reasonably short names for the installation path, profile path, and cell, node, and server names. WebSphere applications tend to have files with long names (for example, com.ibm.test...), and these long names can cause issues with file path length even after profile creation.

1.5.6 Custom profile federation error

If you are creating a custom profile and have selected the option to federate the node during the process (the default) the process must be able to connect to the deployment manager. This means that the deployment manager must be active and the parameters for connection must have been entered correctly. The steps to resolve this problem are:

- Verify the deployment manager is up and running.
- Verify the port is correct.
- Verify the deployment manager user ID and password were specified correctly and the user ID has Administrator authority. This verification is only necessary if administrative security has been enabled for the cell.
- Verify that the deployment manager host can do a host name lookup on the host of the node that is being federated. Use ping to make sure the host name can be found. For example, from the deployment manager, issue the following command:

  ```
  ping newcustomnode.itso.ibm.com
  ```

- Verify that the host of the custom profile node can do a host name lookup on the deployment manager node.
- Verify that the deployment manager system and the new node system have synchronized date and time.
- Recreate the profile, or if the profile was successfully created, federate the node using the **addNode** command.

> **Tip:** If you are having trouble finding the cause of the federation error, create the profile and elect to federate later. Then, after the profile is created, open a command line and use the **addNode** command to federate. The errors will be more visible and retrying the process will be faster.

Start the deployment manager

Do the following to ensure that the deployment manager is available for the federation process:

1. Check the status of the deployment manager, **dmgr**.
2. If the **dmgr** status is not `STARTED`, start it.

Check the status of the deployment manager

Check the status of the deployment manager, **dmgr**, with the following command:

- (Windows): ***dmgr_profile_home*\bin\serverStatus -all**
- (UNIX): ***dmgr_profile_home*/bin/serverStatus.sh -all**
- (i5/OS): WRKSBSJOB SBS(QWAS61). Visually check to see if the **dmgr** is running in that subsystem.

Start the deployment manager

If the **dmgr** status is not `STARTED`, start it with the following command:

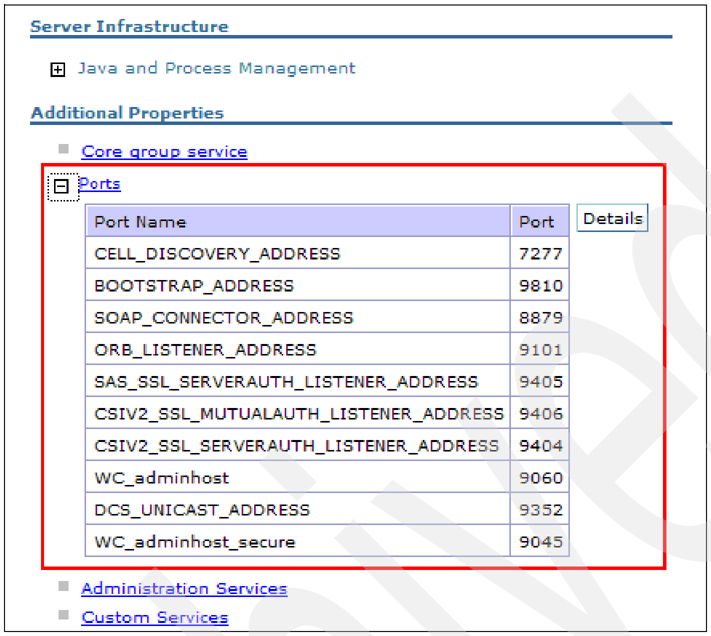
- (Windows): ***dmgr_profile_home*\bin\startManager**
- (UNIX): ***dmgr_profile_home*/bin/startManager.sh**
- (i5/OS): *dmgr_profile_home*/bin/startServer

Verify SOAP port

You can verify the SOAP port using the administrative console or by viewing the portdef.props file. In the administrative console:

1. Select **System administration** → **Deployment manager**.
2. Expand **Ports** to display a table of the ports used.

3. Find the port listed for SOAP_CONNECTOR_ADDRESS. In Figure 1-2, this is port 8879.


Server Infrastructure

⊞ Java and Process Management

Additional Properties

- Core group service
- ⊟ Ports

Port Name	Port
CELL_DISCOVERY_ADDRESS	7277
BOOTSTRAP_ADDRESS	9810
SOAP_CONNECTOR_ADDRESS	8879
ORB_LISTENER_ADDRESS	9101
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9405
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9406
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9404
WC_adminhost	9060
DCS_UNICAST_ADDRESS	9352
WC_adminhost_secure	9045

Details

- Administration Services
- Custom Services

Figure 1-2 Display of the deployment manager ports

To view the port in the portdef.props file:

1. Open *dmgr_profile_home*/properties/portdef.props.
2. Browse the file for the `SOAP_CONNECTOR_ADDRESS` and note the value. Example 1-8 shows a sample portdef.props file.

Example 1-8 The portdef.props file

```
#Generated by PMT GUI
#Thu Mar 08 15:23:41 EST 2007
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=9403
WC_adminhost=9060
DCS_UNICAST_ADDRESS=9352
BOOTSTRAP_ADDRESS=9809
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=9401
CELL_DISCOVERY_ADDRESS=7277
SOAP_CONNECTOR_ADDRESS=8879
ORB_LISTENER_ADDRESS=9100
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=9402
WC_adminhost_secure=9043
```

Verify user ID and password

To federate a profile to a deployment manager with administrative security enabled, you must supply a user ID with the Administrator role. If you received the `ADMN0022E: Access is denied ...` message, you probably have supplied a user ID with something less than Administrator authority.

If you are not sure whether administrative security is enabled, the easiest way to tell is to attempt to access the administrative console.

- If administrative security is turned on, the login panel displays fields for user ID and password. You have to enter a valid user ID and password to login.
- If administrative security is not turned on, there is only a field for the user ID. You can login by specifying any value for the user ID, or you can login without specifying a user.

You can display users with Administrator authority from the administrative console by navigating to **Users and Groups** → **Administrative User Roles**.

Clean up the node after a federation error

If the profile creation fails due to a federation process, you will need to clean up the profile to start over:

1. Use the **`manageprofiles -listProfiles`** command to see whether the profile exists (has been registered to WebSphere). For example, on a Windows system:

   ```
   C:\WebSphere\AppServer\bin>manageprofiles -listProfiles
   [CustomNode01]
   ```

   A response of empty brackets (`[ ]`) means that no profiles exist.

2. If the profile exists, delete it using the process that is outlined in "Deleting profiles" on page 35.

   As an alternative, you can use the profile and federate it manually.

3. If the profile does not exist or you remove it, delete the *profile_home* directory. For example, c:\WebSphere\AppServer\profiles\CustomNode01.

Federate the node manually

If the profile is created, you can use the **`addNode`** command to federate the profile. You run the **`addNode`** command from the *custom_profile_home*/bin directory:

```
addNode dmgr_host soap_port -user Administrator_ID -password password
```

For example:

```
C:\WebSphere\AppServer\profiles\Custom03\bin>addNode ITSOdmgr 8883
-user Administrator -password password
```

1.5.7 Invalid ports selected

For non-root installations on UNIX systems, you should select ports that are higher than 1024 during profile creation.

The installation process warns you with a message indicating that the port is in use but allows the profile to be created successfully. The problem comes when you try to start the server and it fails to start

Resolve the problem

Repeat the profile creation using 1024 or higher for port numbers.

1.5.8 Validate the solution

When a profile creation fails, you need to correct the problem and restart the profile creation. If you were creating the profile as part of the installation and the installation succeeded but the profile creation did not, do not reinstall. Use the profile management tool or the **`manageprofiles`** command to create the profile. If you created a new profile that cannot be used and want to re-create it, use the process that we describe here to delete the old profile first.

Deleting profiles

To delete a profile, do the following:

- If you are removing a custom profile or application server profile that has been federated to a cell:
  a. Stop the application servers on the node.
  b. Remove the node from the cell using the administrative console or the **`removeNode`** command. Removing a node does not delete it, but restores it to its pre-federated configuration that was saved as part of the federation process.
  c. Delete the profile using **`manageprofiles -delete`**.
  d. Use the **`manageprofiles -validateAndUpdateRegistry`** command to clean the profile registry.
  e. Delete the *profile_home* directory.
- If you are removing an application server profile that has not been federated to a cell:
  a. Stop the application server.
  b. Delete the profile using **`manageprofiles -delete`**.

   c. Use the **`manageprofiles -validateAndUpdateRegistry`** command to clean the profile registry.
   d. Delete the *profile_home* directory.

- If you are removing a deployment manager profile:
   a. Remove any nodes federated to the cell using the administrative console or the **`removeNode`** command. Removing a node does not delete it, but restores it to its pre-federated configuration that was saved as part of the federation process.
   b. Stop the deployment manager.
   c. Delete the profile using **`manageprofiles -delete`**.
   d. Use the **`manageprofiles -validateAndUpdateRegistry`** command to clean the profile registry.
   e. Delete the *profile_home* directory.

Deleting a profile with manageprofiles

To delete a profile, use the **`manageprofiles -delete`** command. The format of this command is:

```
manageprofiles -delete -profileName <profile>
```

At the completion of the command, the profile is removed from the profile registry, and the run time components are removed from the *profile_home* directory with the exception of the log files.

If you have errors while deleting the profile, check the following log:

app_server_root/logs/manageprofile/*profile_name*_delete.log

In Example 1-9, we used the **`manageprofiles`** command to delete the profile named Node06.

Example 1-9 Deleting a profile using manageprofiles

```
C:\WebSphere\ND\profiles\Dmgr01\bin>manageprofiles -delete -profileName Node06
INSTCONFSUCCESS: Success: The profile no longer exists.
```

In Example 1-9, it appears that the command executed successfully. However, as an additional step to ensure that the registry was updated properly, you can list the profiles to ensure that the profile has been removed from the registry and validate the registry, as shown in Example 1-10.

Example 1-10 Verifying the delete profile results

```
C:\WebSphere\ND\profiles\Dmgr01\bin>manageprofiles -listProfiles
[Dmgr01, AppSrv01, AppSrv02, SamplesServer, WebServer2Node, DmgrSecure]

C:\WebSphere\ND\profiles\Dmgr01\bin> manageprofiles -validateAndUpdateRegistry
[]
```

Note: If there are problems during the delete, you can delete the profile manually. For information, see *Deleting a profile*, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tpro_removeprofile.html

1.6 Installation Verification Test fails

The Installation Verification Test (IVT) verifies that installation and profile creation were successful. The IVT is the first option on the First Steps console. It can also be run by executing the `ivt` command:

- Windows

  profile_root\bin\ivt.bat
- UNIX

  profile_root/bin/ivt.sh
- i5/OS

  app_server_root/bin/ivt server1 default

  To run this script, your user profile must have *ALLOBJ authority.

The IVT does the following:

- Displays information about the profile
- Starts the server (application or deployment manager depending on the profile)
- Scans the logs for warnings and errors related to the start of the server and displays any of these messages it finds

1.6.1 Identify symptoms

When you run the IVT from the First Steps console, messages indicating the status of the tests and error messages display directly in the console. When running the command, the messages display in the command window. The messages are also logged to the ivtClient and startServer logs.

Symptoms of an IVT failure include:

- IVTL0075I: The Installation Verification Tool verification failed.
- ADMUxxxxE messages
- WSVRxxxxE messages.

Warnings might or might not indicate an error. They are most often useful when they closely precede an error. Example 1-11 illustrates what displays when starting the deployment manager in a successful IVT test.

Example 1-11 Warnings shown in a successful IVT

```
[3/8/07 15:27:30:562 EST] 0000000a WSKeyStore    W   CWPKI0041W: One or
more key stores are using the default password.
[3/14/07 6:24:04:796 EDT] 0000000a WSKeyStore    W   CWPKI0041W: One or
more key stores are using the default password.
[3/8/07 15:27:49:140 EST] 0000000a ThreadPoolMgr W   WSVR0626W: The
ThreadPool setting on the ObjectRequestBroker service is deprecated.
[3/14/07 6:24:11:687 EDT] 0000000a ThreadPoolMgr W   WSVR0626W: The
ThreadPool setting on the ObjectRequestBroker service is deprecated.
IVTL0075I: The Installation Verification Tool verification failed.
```

Port conflicts

The most common cause for problems in the IVT are port conflicts. Processes that are defined by profiles use a set of IP port numbers. For the process (application server, node, or deployment manager) start, these ports must not be in use by other processes.

If you see the `Conflict detected on port` *`xxxx`* message in a log or in the **`ivt`** results, go to "Port conflicts" on page 42.

1.6.2 Collect diagnostics

Collect the following information:

- Profile creation log (see "Profile creation log" on page 46)
- IVT log (see "IVT log" on page 46)
- Start server log (see "Start server log" on page 46)
- SystemOut and SystemErr JVM logs (see "JVM logs" on page 47)

1.6.3 Analyze diagnostics

Note that many error messages are included in multiple logs. Scan these logs in the order that we list here and when you find an indicator of the problem, use the messages to go to the root cause explanation, or to form a search of online support resources.

Analyze profile creation log

If you are not sure whether the profile was successfully created, inspect the profile creation log.

Successful profile creation

Example 1-12 shows messages in the profile creation log that indicate the profile was created successfully. This information can help you rule out a problem with the profile itself being the cause of the IVT failure.

Example 1-12 Messages indicating a successful profile creation

```
<message>INSTCONFSUCCESS: Success: Profile profile_name now exists.
Please consult
app_server_root\profiles\profile_name/logs/AboutThisProfile.txt for
more information about this profile.</message>
```

Profile creation failed

If the profile was not created successfully, search the log for entries with the following text:

- `<level>WARNING</level>`
- `<level>SEVERE</level>`
- `<message> INSTCONFFAILED`
- `<message> INSTCONFPARTIALSUCCESS`
- `<message>text - FAILURE</message>`

If you find INSTCONFFAILED or INSTCONFPARTIALSUCCESS messages go to “Profile creation fails” on page 20.

Analyze the IVT log

The IVT log includes messages from the IVT execution. Search for:

- ADMUxxxxE messages
- Information or warning messages closely preceding an error message.

ADMU3027E, ADMU3028I port conflict messages

Example 1-13 shows how a port conflict would appear in ivtClient.log.

Example 1-13 Port conflict as seen in ivtClient.log

```
>ADMU3028I: Conflict detected on port 8880.  Likely causes: a) An
instance of the server server1 is already running  b) some other
process is using port 8880
>ADMU3027E: An instance of the server may already be running: server1
>ADMU0111E: Program exiting with error:
>           com.ibm.websphere.management.exception.AdminException:
ADMU3027E: An instance of the server may already be running: server1
>ADMU1211I: To obtain a full trace of the failure, use the -trace
option.
>ADMU0211I: Error details may be seen in the file:
>
C:\WebSphere\AppServer\profiles\AppSrv03\logs\server1\startServer.log
IVTL0075I: The Installation Verification Tool verification failed.
```

Root cause: Port conflicts are discussed in “Port conflicts” on page 42.

ADMUxxxxE messages

Review the text and user response information for these messages for possible solutions. You can find the message text and user response information at ADMU messages, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.management.resources.nodeutils.html

Analyze startServer.log

Look for errors that related to the attempt to start the server. Scanning from the top, look for:

- ADMUxxxxE, ADMUxxxxW
- Any other messages ending with `E` (errors) or `W` (warnings).

  Warning messages by themselves do not often indicate a problem. However, warning message in the presence of error messages should be noted.

Review the text and user response information for these messages for possible solutions. You can find the text and user response information for ADMU messages at ADMU messages, which is available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.management.resources.nodeutils.html

Analyze the JVM logs

SystemOut and SystemErr will contain any error messages related to the IVTServlet. The IVT scans SystemOut file looking for errors.

Look for any error or warning messages. Scanning from the top, look for:

- ADMUxxxxE, ADMUxxxxW
- WSVRxxxxE
- Any other messages ending with `E` (errors) or `W` (warnings).

  Warning messages by themselves do not often indicate a problem. However, warning message in the presence of error messages should be noted.

  Review the text and user response information for these messages for possible solutions. You can find the message text and user response information at ADMU messages, which is available at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.management.resources.nodeutils.html
- WSVR messages

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.bootstrap.bootstrap.html

  Note that these messages span three topics in the Information Center.

Invalid port selection in a non-root installation

The following message can occur when ports numbered less than 1024 have been used in a non-root installation:

```
WSVR0009E: Error occurred during startup
com.ibm.ws.exception.RuntimeError: com.ibm.ws.exception.RuntimeError:
org.omg.CORBA.INTERNAL: CREATE_LISTENER_FAILED_4
```

If this is a non-root installation, check the portdef.props file. If you have specified port numbers lower than 1024, delete the profile and re-create it using appropriate port numbers. You can find the portdef.props file at *profile_home*/properties/portdef.props.

Where to go from here:

- If you found evidence of a port conflict during server startup, go to "Port conflicts" on page 42.
- To form a search on any error messages you find, go to "The next step" on page 48.

1.6.4 Port conflicts

The port might already be in use. If you have port conflicts, follow these steps:

1. Look for another running server process that uses the same ports.

   You can find the ports in use by WebSphere processes in *profile_home*/properties/portdef.props.

2. Use the **`netstat`** command to see if the port is in use:
   - UNIX / Linux: **`netstat -an | grep LISTEN`**
   - Windows: **`netstat -an`**. Look for ports in LISTENING state.

Resolve this problem

If the other process will not normally be running (for example, it is a test profile), then bring down that process and retry the IVT.

If there is a chance that the other process will be running at the same time your server will be running, do the following:

1. Delete the profile (see "Deleting profiles" on page 35).
2. Re-create the profile using unique ports.

   The profile management tool can detect ports in use by profiles in that WebSphere installation but not by the profiles in other WebSphere installations or products.

   Use the advanced profile creation option when creating the profile to see the values selected by the profile management tool and to alter any port numbers that have a conflict.
3. Repeat the IVT.

1.6.5 Validate the solution

Retry the IVT. If you get the following message, the IVT has completed correctly:

```
IVTL0070I: The Installation Verification Tool verification succeeded.
```

If necessary, create a new profile after correcting the error, and run the IVT on the new profile.

1.7 Maintenance on i5/OS fails

Fixes for WebSphere Application Server V6.1 for i5/OS are delivered through a fixpack that is included in the WebSphere Application Server for i5/OS group PTF:

- The group PTF number for all WebSphere Application Server V6.1 products for i5/OS release V5R3M0 is SF99322.
- The group PTF number for all WebSphere Application Server V6.1 products for i5/OS release V5R4M0 is SF99323.

After you apply the group PTF, the fixpack is placed under the V6.1 Update Installer directory. You must then run the update script to install the fixpack. This activity discusses what happens when the update script fails.

You can find detailed instructions to complete the fixpack installation in /QIBM/ProdData/WebSphere/UpdateInstaller/V61/UPDI/ReadmeV61.txt.

1.7.1 Identify symptoms

Because the update is run in batch, a failure can only be seen by looking in the logs. The most likely symptom of a failure in the update process is that an error you applied a fix to correct is still occurring.

1.7.2 Collect diagnostics

Collect all the files in the following location:

app_server_root/logs/update/6.1.0-WS-WAS-i5osPPC-FP000000x.install

1.7.3 Analyze diagnostics

Scan each log for indications of an error. Search for the text `error`. Error conditions addressed in this paper are:

- `The server being updated is running`
- `Authority error`
- `Invalid host name or unable to find host name error`
- Host servers are not started

If you find error messages not listed here or you see no messages that indicate the problem, go to “The next step” on page 48.

1.7.4 The server being updated is running

You cannot install a fixpack for WebSphere Application Server V6.1 while the application servers are active. To resolve this issue:

1. Issue the following command:

   WRKSBSJOB QWAS61

2. End any servers that need to be updated.
3. Retry the update.

1.7.5 Authority error

This error is issued when the user who initiated the job does not have enough authority to run the update script. The user initiating the update script must have *ALLOBJ authority.

1.7.6 Invalid host name or unable to find host name error

Option 12 in CFGTCP must match (including case and qualification) option 10 in CFGTCP. For example, if CFGTCP option 12 has *mysystem* as the host name and *IBM.COM* as the domain name, then the correct IP address in CFGTCP option 10 must have *mysystem.IBM.COM* configured as a host name.

1.7.7 Host servers are not started

Issue the STRHOSTSVR *ALL command and retry the update.

1.7.8 Validate the solution

To validate the solution, correct the problem and retry the update.

1.8 Collecting logs

This section provides more information about where to find the diagnostic data for installation problems.

1.8.1 Installation log

You can find the installation log at the following location:

- i5/OS
  - When the installation is performed locally on i5/OS (silent):

    app_server_root/V61/*install_ver*/logs/install/log.txt

    where *install_ver* = Base, ND, Express
  - When the installation is remote to i5/OS from a Windows system:

    user.home\Local Settings\temp\niflogs

    where *user.home* is generally \Documents and Settings\userid (or administrator)

- Distributed platforms

  app_server_root/logs/install/log.txt

  If the installer fails at a very early stage, this log file might not be created or it might exist in the system temporary area, %TEMP%\log.txt in Windows or /tmp/log.txt in UNIX.

Tip: If the installation wizard will not start and there is no installation log, try repeating the installation using a silent install with the `-log` parameter to create the log (for example, Windows):

```
install -options response_file -silent -log # !log_file_name  @ALL
```

1.8.2 Profile creation log

You can find the profile creation log at the following location:

- i5/OS:

  `QIBM/ProdData/WebSphere/AppServerV61/Base/logs/install/createDefaultProfile.log`

- Distributed platforms

  `app_server_root/logs/manageprofiles/profile_name_create.log`

  or

  `app_server_root/logs/manageprofiles/create.log`

1.8.3 Start server log

You can find messages that were issued during the server startup at Distributed and i5/OS platforms, which is available at *profile_root*/logs/*server_name*/startServer.log.s

1.8.4 IVT log

Messages issued during the Installation Verification Test are issued directly to the console and also logged at Distributed platforms, *profile_root*/logs/ivtClient.log.

1.8.5 Profile management tool log

The profile management tool log includes messages that are displayed in the wizard. Collect this log if:

- You used the wizard to attempt the installation and closed it without noting the error messages.
- You are installing on a distributed platform (this log is not available for i5/OS installation).

The log is located at *app_server_root*/logs/manageprofiles/pmt.log.

1.8.6 addNode command log

The **`addNode`** command is used to federate the node defined by the custom profile to the cell. Messages issued during this process are recorded in the addNode log. Collect this log if:

- You are attempting to create a custom profile and
- You have elected to have the node federated during the profile creation process (you did not select **Federate this node later** in the wizard.

You can find the addNode log in the Distributed and i5/OS platforms, *profile_root*/logs/addNode.log.

1.8.7 JVM logs

JVM logs, often referred to as SystemOut and SystemErr logs, are created for every WebSphere Application Server process (application server, cluster member, node agent, and deployment manager). For WebSphere Application Server V6.x (distributed and i5/OS), you can find them in the following locations:

- The JVM log files are by default named SystemOut.log and SystemErr.log. The default location for the SystemOut and SystemErr logs is:
  - *profile_root*/logs/*server_name*/SystemOut.log
  - *profile_root*/logs/*server_name*/SystemErr.log:
- The location of application server logs is configurable.
  a. Select **Troubleshooting** → **Logs and Trace** in the navigation bar.
  b. Click on the server name.
  c. Select **JVM logs**

  This page shows location of the log file.

1.9 The next step

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, other issues can occur during installation.

1.9.1 Search online support

If you are sure the problem is in the installation process, there are tasks that you can do before contacting IBM support.

First, you should review the documentation that you have gathered for errors that were not addressed in this paper and search support sites for information or fixes. Look for current information available from IBM support on known issues and resolutions on the following IBM support page:

http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCVS24

Look also at the WebSphere Information Center *Troubleshooting installation* documentation for additional resources for diagnosing and fixing installation issues:

- Network Deployment on distributed platforms:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tins_trouble.html

- Network Deployment on i5/OS

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.iseries.doc/info/iseriesnd/ae/tins_trouble.html

1.9.2 Contact IBM

If these steps do not resolve your problem, then gather additional information as specified in the following MustGather document and raise a problem record (PMR) with IBM. The following URL contains a list of the MustGather documentation for installation problems.

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21255887

Part 2

Web container problem determination

This part contains the following IBM Redpaper:

- *WebSphere Application Server V6.1: Web Container Problem Determination*, REDP-4309 by Carla Sadtler, Giribabu Paramkusham, and Robert Larsen

2

Web container problem determination

The runtime environment for Web components is called the *Web container*. If users receive unexpected results in a Web browser (such as errors or incorrect information), you might have a problem with the Web container.

Potential symptoms of a Web container problem can include:

- Users are not able to access a Web component.
- There is unexpected behavior when running a Web component.
- Errors occur when a Web component is started.
- There are problems with JSP™ compilation.
- Errors or exceptions are thrown during the running of a Web component.
- Messages that start with SRVE (Web container), JSPG (JSP), or JSFG (JSF pages) are received.

This paper takes you through the steps of diagnosing Web container problems for IBM WebSphere Application Server V6.1 on distributed and IBM i5/OS platforms.

2.1 Introduction to Web containers

A Web container is a runtime environment for Web applications. It processes servlets, JSP files, and other types of server-side components. Each application server runtime has one logical Web container, which can be modified but not created or removed.

2.1.1 Web container overview

Figure 2-1 illustrates the Web container and its place in the application server.

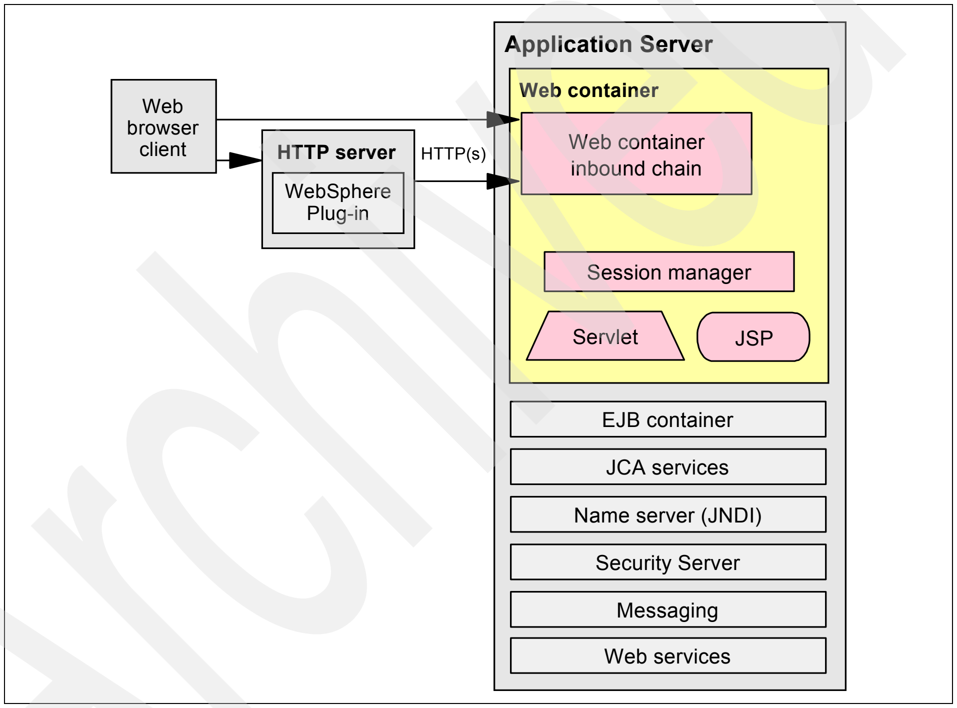


Figure 2-1 Web container overview

Each Web container provides:

- Web container transport chains

  The Web container inbound transport chain handles requests. It consists of a TCP inbound channel that provides the connection to the network, an HTTP inbound channel that serves HTTP 1.0 and 1.1 requests, and a Web container channel that sends requests for servlets and JSPs to the Web container for processing.

- Servlet processing

  When it handles servlets, a Web container creates a request object and a response object and then invokes the servlet service method. The Web container also invokes the servlet destroy method when appropriate and unloads the servlet, after which the JVM performs garbage collection.

- HTML and other static content processing

  Requests for HTML and other static content that are directed to the Web container are served by the Web container inbound chain. However, in most cases, using an external Web server and a Web server plug-in as a Web container interface is more appropriate for a production environment.

- Session management

  The Web container provides support for the javax.servlet.http.HttpSession interface as described in the API specification for the servlet.

2.1.2 Web applications

Servlets and JSP files are referred to as Web components. Static content files (such as HTML pages, image files, and XML files) are bundled with Web components during application assembly to create a Web module. A Web module is the single deployable and usable unit of Web resources and has a specific structure or archived format known as a *Web archive* (WAR) file. A J2EE Web module corresponds to a Web application as defined in the Java servlet specification.

The top-level directory of a Web module is the *document root* of the application. The document root is where JSP pages and static Web resources are stored. The document root has a subdirectory that is named /WEB-INF/, which contains the Web application deployment descriptor (web.xml) and the server-side classes used by the module.

Figure 2-2 on page 54 illustrates the directory structure of a Web application.

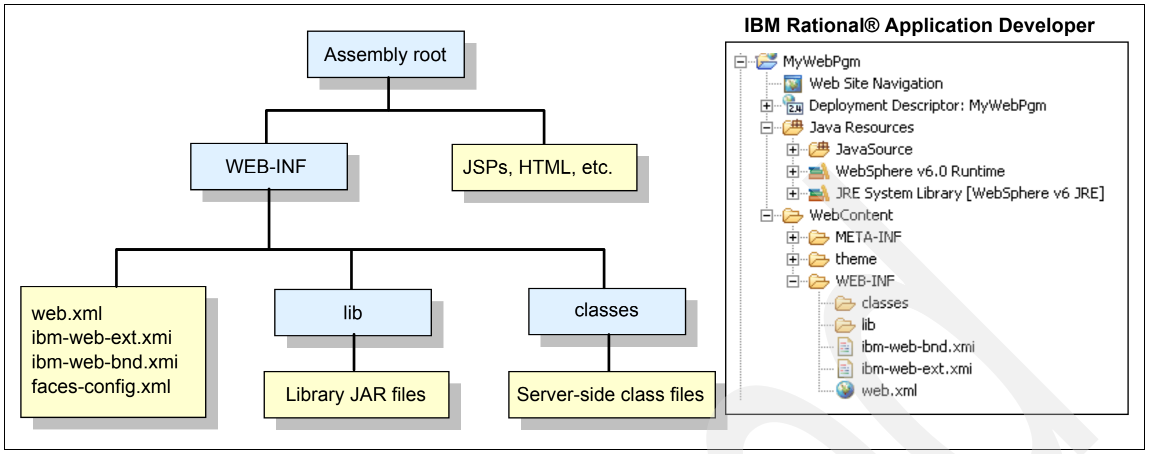


Figure 2-2 Web application components

2.2 Determine the Web container problem type

Many indicators of a Web container problem first appear to a user as an unexpected Web browser page or page contents, such as:

- HTTP 404 errors
- HTTP 500 errors
- Incorrect information displayed on Web pages

The unexpected information can be an error page that results from an HTTP error or a page that displays incorrect or incomplete information.

When an HTTP error occurs, the Web browser displays the error page with the error code. If a custom error page has been configured for the Web module, you must check the Web server log files to determine the HTTP error code because the true error might not be apparent.

Where to go from here: If you know the type of error, you can go directly to:

- “HTTP 404 errors” on page 55
- “HTTP 500 errors” on page 75
- “Incorrect information displayed on Web pages” on page 85

If you do not know the type of error, continue with“Collect and analyze the Web server log” on page 55.

2.2.1 Collect and analyze the Web server log

Collect the access log for the Web server. The Web server log file names and locations are specific to the product.

To analyze the log, search the Web server access log for 404 or 500 errors.

The following examples are from IBM HTTP Server logs:

- From the access.log:

  ```
  127.0.0.1 - - [28/Mar/2007:19:52:31 -0400] "GET url HTTP/1.1" 404 304
  127.0.0.1 - - [28/Mar/2007:20:03:48 -0400] "GET urlHTTP/1.1"500 5348
  ```

  Note the URL the Web server was trying to serve when the error occurred.

- If the Web server expected to serve the URL (versus passing the request to an application server), there is a corresponding error with information about where the Web server expected to find the file. Search the Web server error log for errors indicating that a file was not found, such as:

  ```
  - [Wed Mar 28 19:52:31 2007] [error] [client 127.0.0.1] File does
  not exist: file_location
  ```

2.3 HTTP 404 errors

HTTP 404 errors can have different underlying causes. Some examples of these causes are:

- External factors, such as a problem in the Web server
- Configuration problems, such as an incorrect Web server plug-in or virtual host configuration
- Runtime problems, such as an application or application server not started
- User or application problems, such as an incorrectly specified URL

2.3.1 Check system integrity

When users receive HTTP 404 error codes and you know the application at the root of the problem, the first thing to check is the integrity of the following system components:

- Web server
- Application server
- Application

If these are all working properly or you are not sure which application and server are involved, collecting more information about the symptoms is necessary.

Verify that the Web server is responding

Verify that the Web server is responding to requests. How you do this depends on your Web server product and its configuration.

If you are using IBM HTTP Server, a quick test is to access the URL for the Web server from a Web browser:

```
http://server_name
```

If you see the Welcome page, the Web server is running and responding to requests. If the Welcome page does not appear (that is, if you get a browser message such as page cannot be displayed or something similar), the problem is most likely in the Web server.

Resolve Web server problems

For general information about how to approach Web server problems, see “Approaching problems with Web servers and plug-ins” on page 100.

Verify that the application server is started

To verify the health of the hosting application server:

1. Determine the server or cluster that hosts the application.
2. Check the status of the application server.
3. Start the application server if it is not running.

If you are not sure about how to follow these steps, see “Managing servers and applications” on page 103.

Verify that the application is started

The status of an application can be seen in the administrative console. Select **Applications** → **Enterprise Applications**.

A Started status means the application is running, but it does not indicate if there were problems during the application startup.

If the application is not started, attempt to start it by selecting the application and clicking **Start**.

2.3.2 Collect diagnostics

If the type of error is not apparent to you or you did not obtain the URL information, collect the following diagnostic data:

- Errors displayed by the user browser, including the URL that failed
- SystemOut log data for the application server

  If your application is deployed to a cluster, you might need to collect the logs from each active server in the cluster. See "JVM logs" on page 97.
- Web server access and error logs

  If you are running an IBM HTTP Server, see "IBM HTTP Server, HTTP Server (powered by Apache) log files" on page 98.
- TRCTCPAPP trace (i5/OS)

  Because the HTTP Server (powered by Apache) is integrated into i5/OS, there is additional tracing available that can provide useful information. See "Trace Web server requests to WebSphere (i5/OS)" on page 98 for information about collecting this trace.

2.3.3 Analyze diagnostics

The two key determinations that you can make from the diagnostic data are:

- The URL that failed
- Where the failure occurred (Web server, plug-in, application server)

Analyze the diagnostics in the following order:

1. The browser error display

   The errors displayed by the browser can indicate the failure type and URL that failed. This might be enough information to resolve the problem.
2. SystemOut

   If the failure occurs in the Web container, SystemOut has the most informative error messages. These might be displayed by the browser also, depending on the error handling of the application. If no error messages are seen in the JVM logs, the request most likely did not reach the application server.
3. The TRCTCPAPP trace (i5/OS only)

   This trace helps you determine if the problem is caused by the plug-in.
4. Web server log

   If no other error indications are found, look at the Web server log, which can tell you the type of error (404 or 500) and the URL that caused the error.

2.3.4 Analyze the errors displayed by the browser

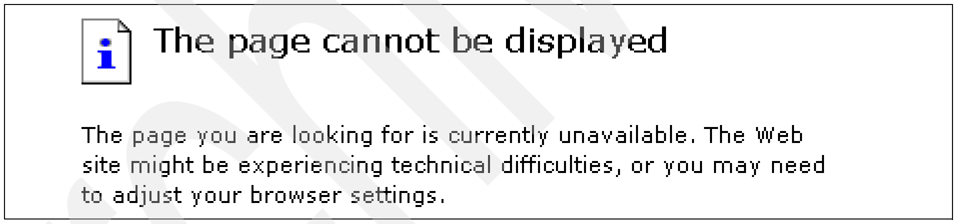
HTTP 404 errors often appear in the client browser with descriptive text and possibly with additional error information. If the following errors have been reported by users and you know the failing URL, you can determine the root cause of the problem:

- The page cannot be displayed
- JSP error or JSF error
- Failed to find resource
- File not found
- WebGroup/virtual host not defined

The type of error is not always be apparent to the user. Web applications often intercept and handle these types of errors, presenting a generic error page to the Web client. In that case, check the application server and Web server logs.

The page cannot be displayed or found

Figure 2-3 shows a typical example of a Web page with this error message. The display varies, depending on the browser; for example, instead of this page, you might see “The page cannot be found.”



The page cannot be displayed

The page you are looking for is currently unavailable. The Web site might be experiencing technical difficulties, or you may need to adjust your browser settings.

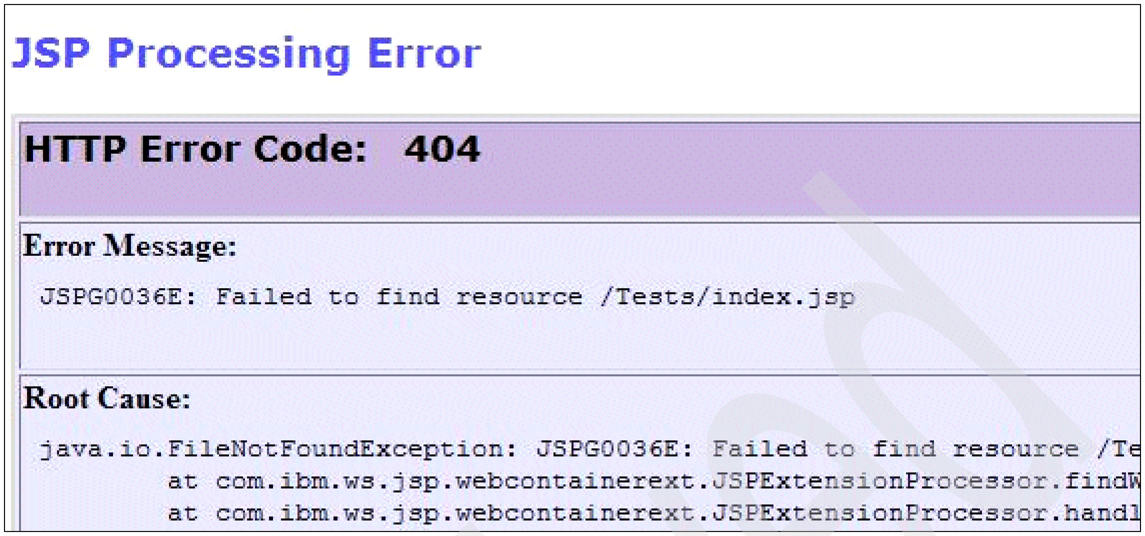
Figure 2-3 HTTP 404 Page cannot be displayed

Possible root causes for this error include a URL that was specified incorrectly and the unavailability of a component that is required to access the page (Web server, application server, for example). Note the URL in the browser at the time the error is displayed.

If the resource is a JSP or servlet, see “Page cannot be displayed or JSP/JSF error” on page 66.

JSP and JSF errors

JSP and JSF errors are usually accompanied by a message from the application server that indicates the problem. Figure 2-4 on page 59 shows a JSP error with a "JSPG0036E: Failed to find resource” message.



JSP Processing Error

HTTP Error Code: 404

Error Message:

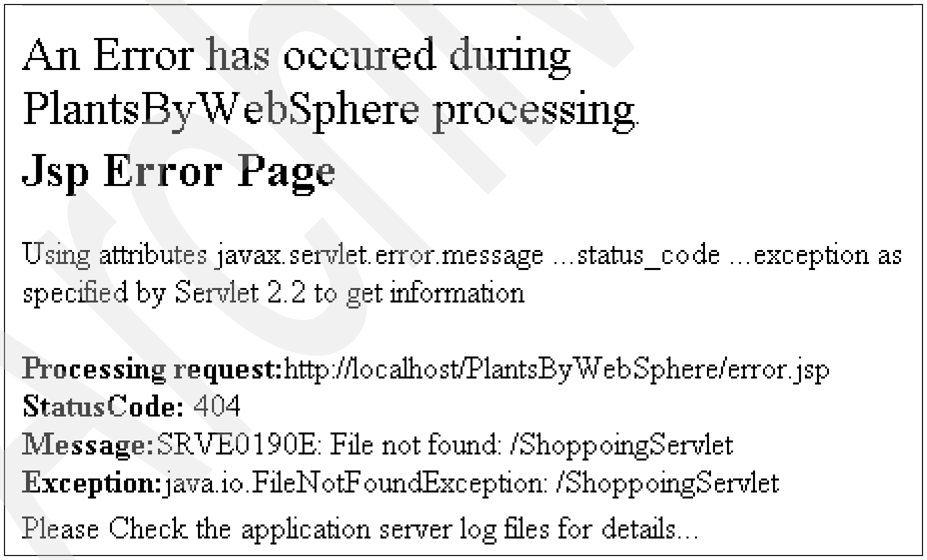
```
JSPG0036E: Failed to find resource /Tests/index.jsp
```

Root Cause:

```
java.io.FileNotFoundException: JSPG0036E: Failed to find resource /Te
        at com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor.findW
        at com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor.handl
```

Figure 2-4 HTTP 404 JSP error, JSPG0036E

Figure 2-5 shows a JSP error with an "SRVE0190E: File not found" message.



An Error has occured during PlantsByWebSphere processing.

Jsp Error Page

Using attributes javax.servlet.error.message ...status_code ...exception as specified by Servlet 2.2 to get information

Processing request:http://localhost/PlantsByWebSphere/error.jsp
StatusCode: 404
Message:SRVE0190E: File not found: /ShoppoingServlet
Exception:java.io.FileNotFoundException: /ShoppoingServlet
Please Check the application server log files for details...

Figure 2-5 JSP error, SRVE0190E

Possible root causes for this error include an incorrectly specified URL, the page is not available on the server, and a Web server plug-in configuration problem.

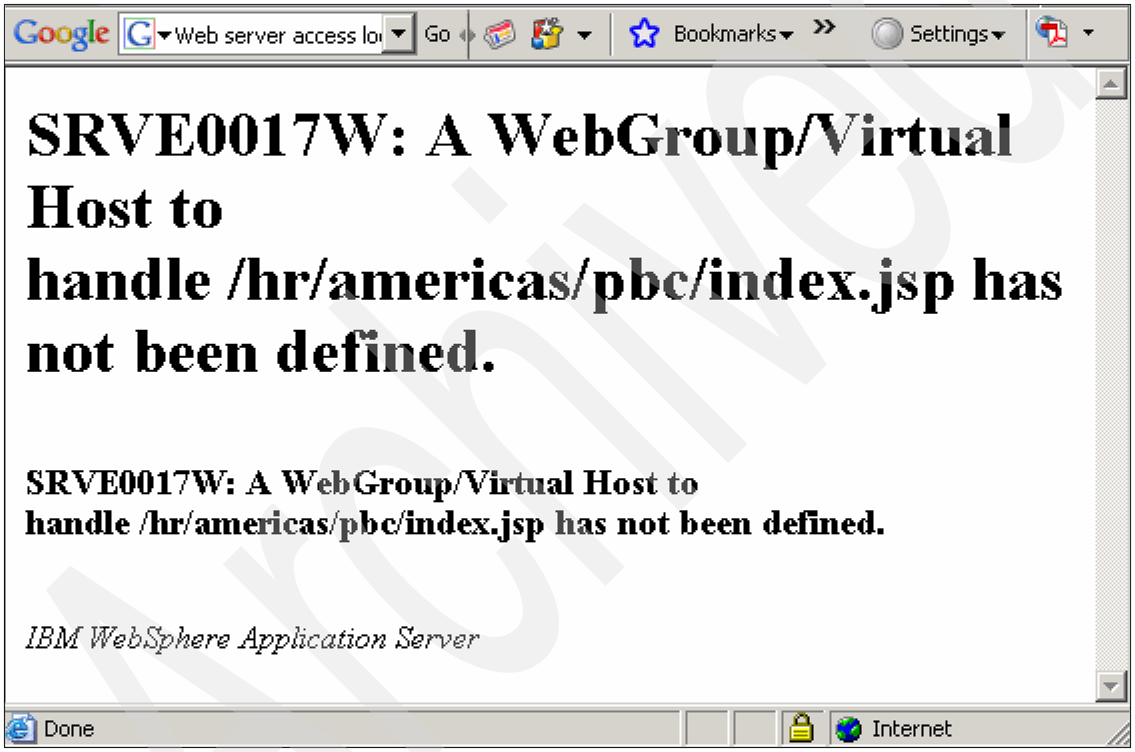
Note the URL that caused the error. For more information, see “Page cannot be displayed or JSP/JSF error” on page 66.

WebGroup/Virtual Host has not been defined

The primary reasons for this error are virtual host configuration errors and an incorrectly specified URL. It is usually accompanied by the following messages:

- `SRVE0255E: WebGroup/Virtual Host has not been defined` (V6.1)
- `SRVE0017W: WebGroup/Virtual Host has not been defined` (V6.0)

Figure 2-6 is an example of what the SRVE0017W message looks like.



SRVE0017W: A WebGroup/Virtual Host to handle /hr/americas/pbc/index.jsp has not been defined.

SRVE0017W: A WebGroup/Virtual Host to handle /hr/americas/pbc/index.jsp has not been defined.

IBM WebSphere Application Server

Figure 2-6 WebGroup/Virtual Host failure

Note the URL that caused the error. For more information, see “WebGroup/virtual host not defined” on page 66.

2.3.5 Analyze SystemOut

To analyze SystemOut, search for:

- Web container messages: SRVExxxxE or SRVExxxxW messages
- JSP messages: JSPGxxxxE or JSPGxxxxW messages
- JSF messages: JSFGxxxxE or JSFGxxxxW messages
- Error messages that are related to the application startup

Check for successful application start

The sequence of messages in Example 2-1 shows a normal application and Web module startup sequence.

Example 2-1 *Normal application startup messages*

```
ApplicationMg A   WSVR0200I: Starting application: [application_name]
WebContainer  A   SRVE0161I: IBM WebSphere Application Server - Web 
Container.
Copyright IBM Corp. 1998-2004
WebContainer  A   SRVE0162I: Servlet Specification Level: 2.4
WebContainer  A   SRVE0163I: Supported JSP Specification Level: 2.0
WebGroup      A   SRVE0169I: Loading Web Module: [web_module_name]
ApplicationMg A   WSVR0221I: Application started: [application_name]
```

If you received error or warning messages during application startup, you must address the problem with the application.

Web group/virtual host errors

If you see one of the following messages, you might have a problem with the virtual host configuration:

- `SRVE0255E: A WebGroup/Virtual Host to handle` *`url`* `has not been defined.` (V6.1)
- `SRVE0017W: A WebGroup/Virtual Host to handle` *`url`* `has not been defined.` (V6.0)

For more information, see “WebGroup/virtual host not defined” on page 66.

Other error messages

Error messages with JSPG, JSFG, or SRVE prefixes have information about the error, including the URL that caused the problem. Note the URL and see “Page cannot be displayed or JSP/JSF error” on page 66.

Where to go from here: If you found an error message that is not in the list, see “The next step” on page 99. If you did not find any errors, see “Analyze the Web server logs” on page 62.

2.3.6 Analyze the Web server logs

You can search for error indications in the Web server logs and analyze them.

Search for 404 status codes in the access log

A log entry in the NCSA format looks like this:

```
Remote_host- - date_time "request" status_code bytes
```

For 404 errors:

1. Search for "404" in *status_code*.
2. Note the *remote_host* address, the *date_time* stamp, and the *request* URL.

Search for errors in the error log

An error entry in the error log resembles this example:

```
[date_time] [error] [client Remote_host] error_message
```

For errors in this entry:

1. Search for "error."
2. Correlate the access log entry that you noted with the error log entry by comparing the *date_time* and *remote_host* entries.
3. Note the *error_message* text.

Evaluate the access log

HTTP errors that are displayed by the Web browser are logged in the access log. These errors can originate in the Web server or application server.

The primary value of analyzing the access log entry is to determine the failing URL.

Static resources (for example, HTML pages and images) are normally served by the Web server. However, requests for static resources are sent to the application server when the fileServingEnabled property is set to **true** in the Web module extended deployment descriptor, which is ibm-web-ext.xmi. When an error occurs during the process of serving static resources from the application server, the only indication of the failure is in the access log.

If you see a 404 error in the access code, but no corresponding error in the Web server error log or SystemOut, check the fileServingEnabled property. If it is set to **true**, verify that the resources exist in the proper path in the application server.

Evaluate the error log

The presence of a corresponding message in the error log indicates that the Web server attempted to handle the request and failed (as opposed the WebSphere Application Server). The message contains more information about the error, including the file that the Web server was trying to serve. This indicates that the problem is in the Web server.

This section provides some examples of error messages.

> **Note:** If there is a 404 error in the access log, file serving is not enabled, and no error is seen in the error log, see "Page cannot be displayed or JSP/JSF error" on page 66 for more information.

Example 1

If errors similar to those that are shown here appear in the Web server logs, they indicate that the URL /sdfsdf.dsf was not found.

- access.log:

  ```
  127.0.0.1 - - [28/Mar/2007:19:52:31 -0400] "GET /sdfsdf.dsf HTTP/1.1"
  404 304
  ```

- error.log

  ```
  - [Wed Mar 28 19:52:31 2007] [error] [client 127.0.0.1] File does
  not exist: C:/IBM/IBM HTTP Server/htdocs/en_US/sdfsdf.dsf
  ```

The message in the error log tells you what file the URL translates to.

The root of this problem is that the file does not exist in the Web server. Verify that the URL is correct and make sure that the file is in the proper location on the Web server.

Example 2

If you see an error similar to the one that is shown here in the access log, but no corresponding error in the error log, it indicates that the Web server did not consider this a request that it should handle:

```
127.0.0.1 - - [28/Mar/2007:19:54:49 -0400] "GET /HitCount2.jsp
HTTP/1.1" 404 2876
```

The most likely cause is a problem with the Web server plug-in or at the application server.

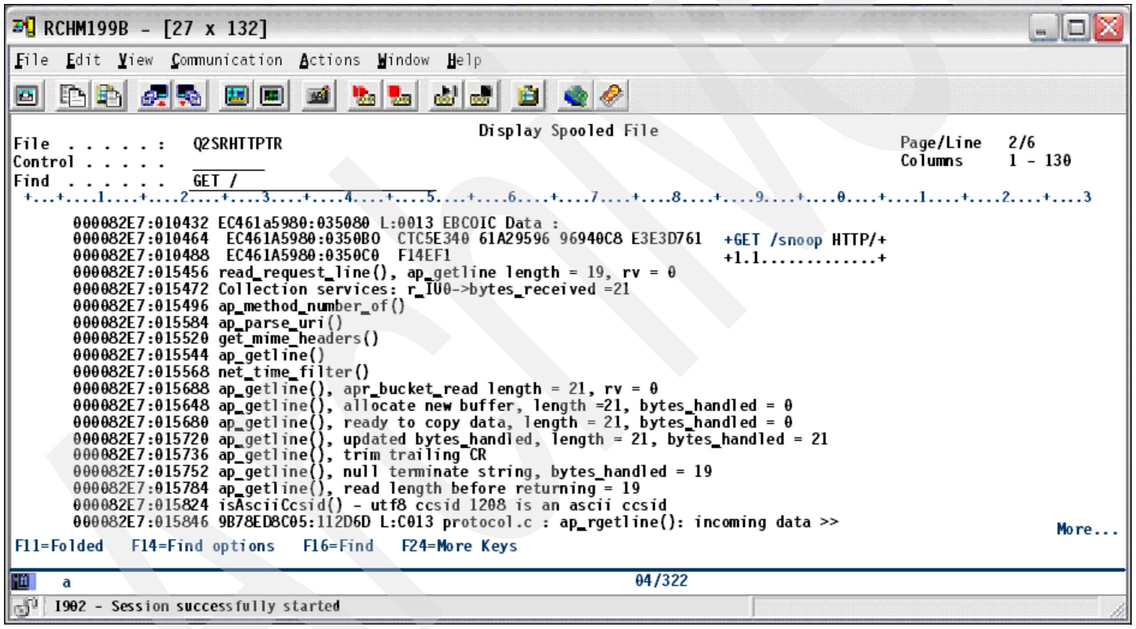
Verify that the plug-in configuration is correct (see "Verify that the Web server plug-in is working correctly" on page 71) and that the URL is correct (see "Verify that the URL is correct" on page 67).

Where to go from here: If you still feel sure that you have a Web container problem, but your symptoms are not listed here, go to "The next step" on page 99 to find information about searching online support and preparing to contact IBM if necessary.

2.3.7 Analyze the TRCTCPAPP trace

If your problem occurred in i5/OS and you collected this trace, analyze the trace contents:

1. Display the spool file.
2. On the Find line at the top, type **GET /** and press **F16** to locate the first request in the trace (Figure 2-7).



```
                                   Display Spooled File
File  . . . . . :   Q2SRHTTPTR                                                    Page/Line   2/6
Control . . . . .   ________                                                      Columns     1 - 130
Find  . . . . . .   GET /
 +...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....0....+....1....+....2....+....3
      000082E7:010432 EC461a5980:035080 L:0013 EBCDIC Data :
      000082E7:010464  EC461A5980:0350B0  C7C5E340 61A29596 96940C8 E3E3D761   +GET /snoop HTTP/+
      000082E7:010488  EC461A5980:0350C0  F14EF1                               +1.1.............+
      000082E7:015456 read_request_line(), ap_getline length = 19, rv = 0
      000082E7:015472 Collection services: r_IU0->bytes_received =21
      000082E7:015496 ap_method_number_of()
      000082E7:015584 ap_parse_uri()
      000082E7:015520 get_mime_headers()
      000082E7:015544 ap_getline()
      000082E7:015568 net_time_filter()
      000082E7:015688 ap_getline(), apr_bucket_read length = 21, rv = 0
      000082E7:015648 ap_getline(), allocate new buffer, length =21, bytes_handled = 0
      000082E7:015680 ap_getline(), ready to copy data, length = 21, bytes_handled = 0
      000082E7:015720 ap_getline(), updated bytes_handled, length = 21, bytes_handled = 21
      000082E7:015736 ap_getline(), trim trailing CR
      000082E7:015752 ap_getline(), null terminate string, bytes_handled = 19
      000082E7:015784 ap_getline(), read length before returning = 19
      000082E7:015824 isAsciiCcsid() - utf8 ccsid 1208 is an ascii ccsid
      000082E7:015846 9B78ED8C05:112D6D L:C013 protocol.c : ap_rgetline(): incoming data >>
                                                                                                                    More...
F11=Folded   F14=Find options   F16=Find   F24=More Keys

    a                                                     04/322
I902 - Session successfully started
```

Figure 2-7 Trace data

The column on the right with request information is called the "eye-catcher." The data is surrounded by "*" to help it stand out; for example:

```
*GET /snoop HTTP/*
*1.1.............*
```

3. Scroll down to see additional HTTP header information in the request in the eye-catcher.

4. The following entry type indicates that a request was passed to WebSphere:

   ```
   mod_was_ap20_http: as_translate_name: WebSphere will handle: /snoop
   ```

   This line tells you that the HTTP server matched the incoming request to the rules in the Web server plug-in configuration file and sent the request to WebSphere for processing.

5. Continue to scroll down until you see the outgoing HTTP response to the request.

   A successful response looks like that in Example 2-2.

Example 2-2 *Successful response from WebSphere*

```
*HTTP/1.1 200 OK.*
*.Date: Thu, 05 A*
*pr 2007 14:54:00*
* GMT..Server: We*
*bSphere Applicat*
*ion Server/6.1..*
```

If the request is unsuccessful, the Server header on the outgoing HTTP response tells you whether HTTP or WebSphere is throwing the error.

Example 2-3 is a sample of a 404 response sent by WebSphere that indicates that the plug-in sent the request to WebSphere but the application could not be found on the WebSphere side:

Example 2-3 *HTTP 404 response from WebSphere*

```
*HTTP/1.1 404 Not*
* Found..Date: Th*
*u, 05 Apr 2007 1*
*5:16:49 GMT..Ser*
*ver: WebSphere A*
*pplication Serve*
*r/6.1..........*
```

Example 2-4 shows an outgoing HTTP response that was sent from the HTTP server.

Example 2-4 *HTTP 404 response from the HTTP server to the user*

```
*HTTP/1.1 404 Not*
 * Found..Date: Th*
 *u, 05 Apr 2007 1*
 *5:35:56 GMT..Ser*
 *ver: Apache.....*
```

If the plug-in correctly sent the request to WebSphere, but an error occurred at the server, see “Page cannot be displayed or JSP/JSF error” for more information. If the plug-in did not send the request to WebSphere, but it should have, see “WebGroup/virtual host not defined” for more information.

2.3.8 Root causes for HTTP 404 errors

Based on symptoms that users report, you can usually narrow the problem down to one of two types:

- Page cannot be displayed or JSP/JSF error
- WebGroup/virtual host not defined

Determining the root cause of each type involves a series of checks that ensure that the components of the system (Web server, plug-in, etc.) are working properly and that there are no configuration problems. Some of these system checks are performed for more than one problem type.

Page cannot be displayed or JSP/JSF error

When you have a “Page cannot be displayed” error and the resource is a JSP page, JSF page, or servlet, or the browser is displaying a JSP error, take the following actions:

- Verify that the Web server plug-in is working correctly (see “Verify that the Web server plug-in is working correctly” on page 71).
- Verify that the URL that is causing the error is specified correctly (see “Verify that the URL is correct” on page 67).
- Verify that the Web server is responding (see “Verify that the Web server is responding” on page 56).
- Verify that the application is running (see “Verify that the application is started” on page 56).

If these actions do not resolve your problem, see “The next step” on page 99.

WebGroup/virtual host not defined

For a “WebGroup/virtual host not defined” error, take the following actions:

- Verify that the virtual host configuration is correct (see “Verify that the virtual host configuration is correct” on page 73).
- Verify that the URL causing the error is specified correctly (see “Verify that the URL is correct” on page 67).
- Verify that the application is running (see “Verify that the application is started” on page 56).

If these actions do not resolve your problem, see “The next step” on page 99.

2.3.9 Verify that the URL is correct

In a new installation or new application, it is possible that the URL of the application is not being specified correctly. To determine the URL of the installed application, use the administrative console to view the configuration of a number of items.

The format of the URL is as follows:

```
http://host:port/context_root/servlet_url_pattern
```

To find the URL for a servlet or JSP:

1. Find the URL pattern for the servlet.
2. Find the context root of the Web module that contains the servlet.
3. Find the virtual host where the Web module is installed.
4. Find the aliases for the virtual host.
5. Compare the failing URL with the correct URL for the servlet.

The examples in this section use the ShoppingServlet of the PlantsByWebSphere sample application.

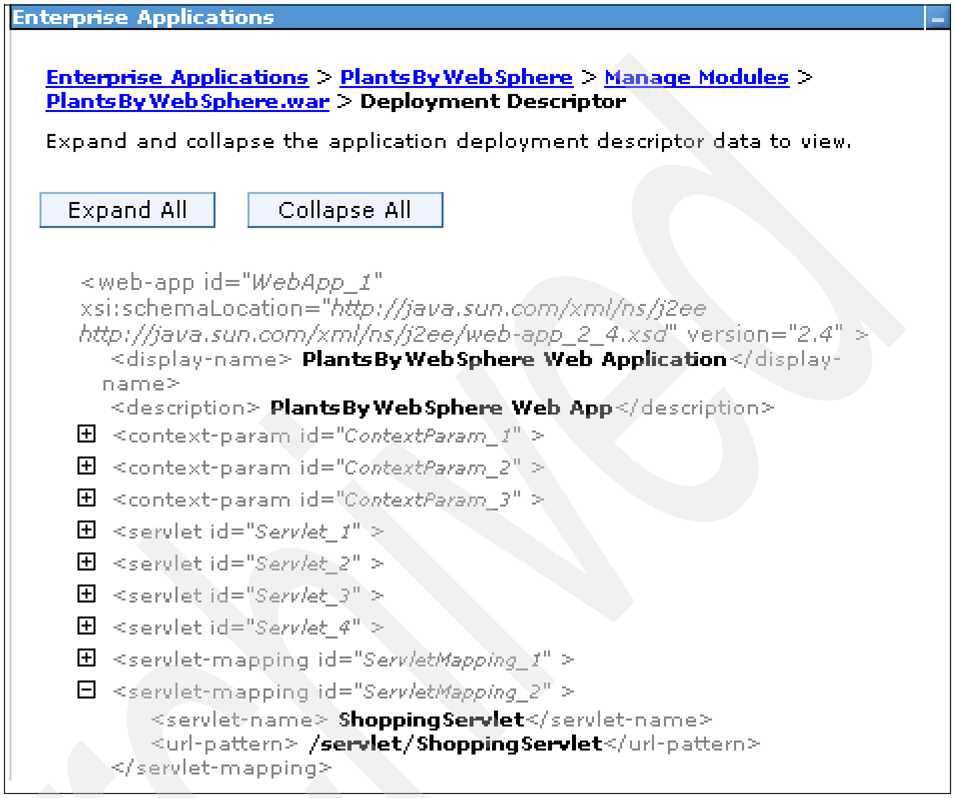
Find the URL pattern (*servlet_url_pattern)*

Find the URL pattern for the servlet in the deployment descriptor of the Web module as follows:

1. In the administrative console, select **Applications** →**Enterprise Applications**.
2. Click the application name to open the Configuration page for the application.
3. Select **Manage Modules** to list the Web and EJB™ modules in the application. If there is more than one Web module, view the deployment descriptor for each until you find the servlet.
4. Click ***Web_module_name*** **Web application** to see the general properties.
5. Click **View Deployment Descriptor**.

   This opens the Web module properties window. Look for the servlet in a <servlet-mapping> entry and note the **url-pattern** value.
6. If the Web module has security configured, check the *<security-constraint>* and *<security-role>* deployment descriptor tags for the role that is needed for access to the selected Web resource.

Figure 2-8 shows the portion of the PlantsByWebSphere Web module deployment descriptor that includes the ShoppingServlet mapping. From this, you can see that the URL pattern is **/servlet/ShoppingServlet**.



Enterprise Applications > PlantsByWebSphere > Manage Modules > PlantsByWebSphere.war > Deployment Descriptor

Expand and collapse the application deployment descriptor data to view.

```
<web-app id="WebApp_1"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4" >
   <display-name> PlantsByWebSphere Web Application</display-
  name>
   <description> PlantsByWebSphere Web App</description>
⊞ <context-param id="ContextParam_1" >
⊞ <context-param id="ContextParam_2" >
⊞ <context-param id="ContextParam_3" >
⊞ <servlet id="Servlet_1" >
⊞ <servlet id="Servlet_2" >
⊞ <servlet id="Servlet_3" >
⊞ <servlet id="Servlet_4" >
⊞ <servlet-mapping id="ServletMapping_1" >
⊟ <servlet-mapping id="ServletMapping_2" >
      <servlet-name> ShoppingServlet</servlet-name>
      <url-pattern> /servlet/ShoppingServlet</url-pattern>
   </servlet-mapping>
```

Figure 2-8 PlantsByWebSphere Web module deployment descriptor

The URL for the example now looks like:

```
http://host:port/context_root/servlet/ShoppingServlet
```

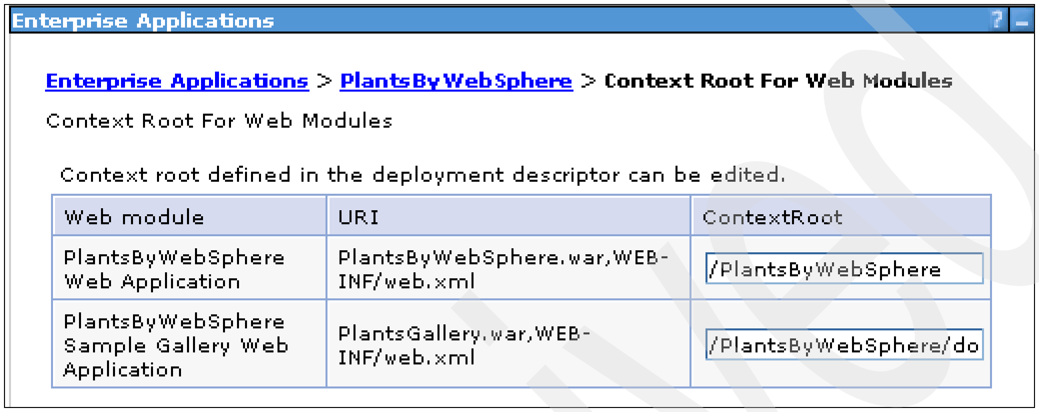
Find the context root (*context_root*)

To find the context root for the URL:

1. Select **Applications** →**Enterprise Applications**.
2. Click the application name.
3. Click **Context Root for Web Modules**.
4. Note the context root for the appropriate Web module.

In Figure 2-9, you can see that:

- There are two Web modules in the PlantsByWebSphere enterprise application. From the previous step, you know the Web module that contains the ShoppingServlet is the PlantsByWebSphere Web module.
- The context root for the PlantsByWebSphere Web module is /PlantsByWebSphere.



Enterprise Applications

Enterprise Applications > PlantsByWebSphere > Context Root For Web Modules

Context Root For Web Modules

Context root defined in the deployment descriptor can be edited.

Web module	URI	ContextRoot
PlantsByWebSphere Web Application	PlantsByWebSphere.war,WEB-INF/web.xml	/PlantsByWebSphere
PlantsByWebSphere Sample Gallery Web Application	PlantsGallery.war,WEB-INF/web.xml	/PlantsByWebSphere/do

Figure 2-9 Context root for the Web modules in DefaultApplication

When you put the context root together with the resource name, you obtain the following result:

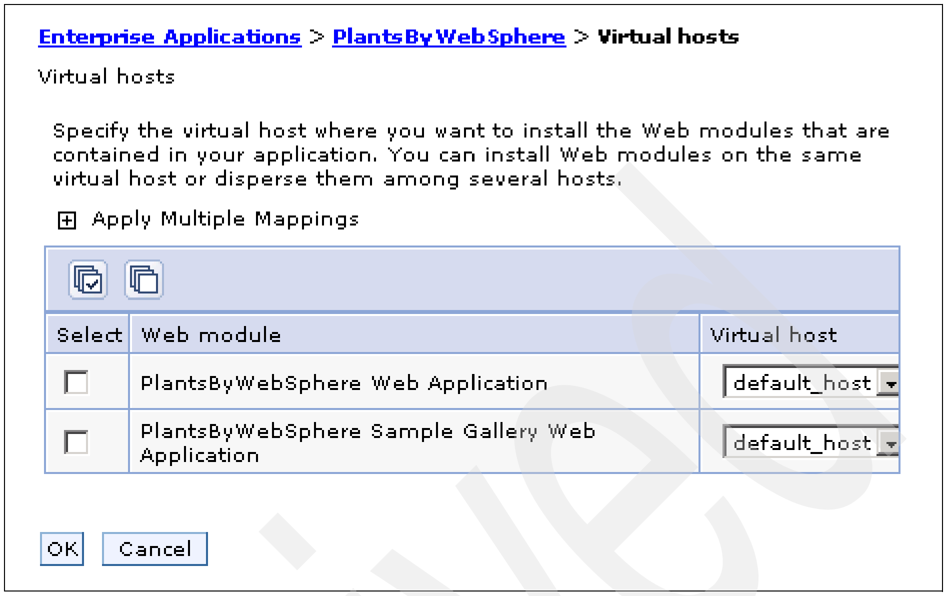
```
http://host:port/PlantsByWebSphere/servlet/ShoppingServlet
```

Find the virtual host (*host:port*)

To find the virtual host where the Web module is installed:

1. Select **Applications →Enterprise Applications**.
2. Click the application name.
3. Click **Virtual hosts** under Web Module Properties to see the Web modules in the application and the virtual hosts in which they have been installed.
4. Note the virtual host for the Web module.

Figure 2-10 on page 70 shows the Web modules in the PlantsByWebSphere application and the virtual hosts in which they have been installed. Note that the PlantsByWebSphere Web module has been installed in *default_host*.


Enterprise Applications > PlantsByWebSphere > **Virtual hosts**

Virtual hosts

Specify the virtual host where you want to install the Web modules that are contained in your application. You can install Web modules on the same virtual host or disperse them among several hosts.

⊞ Apply Multiple Mappings

Select	Web module	Virtual host
☐	PlantsByWebSphere Web Application	default_host
☐	PlantsByWebSphere Sample Gallery Web Application	default_host

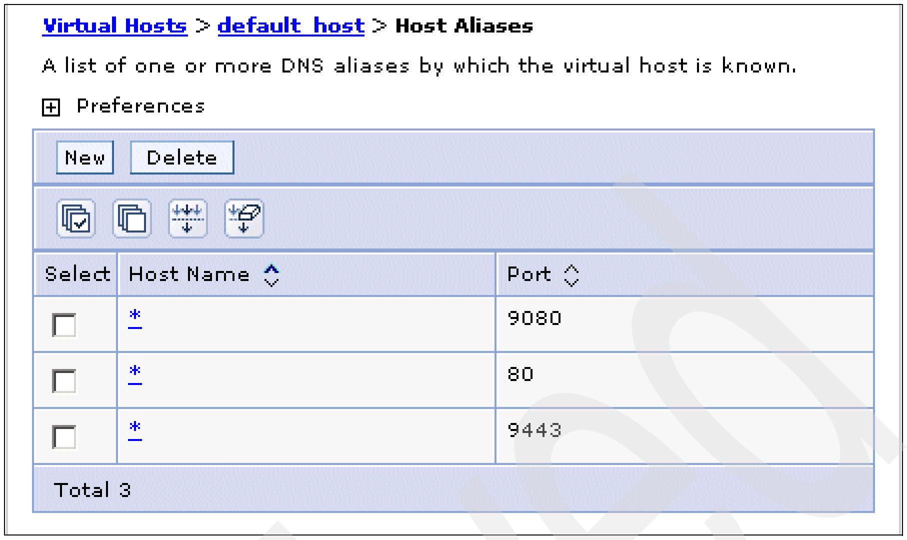
Figure 2-10 List of virtual hosts

Find the aliases for the virtual host

To find the host aliases for the virtual host:

1. From the console navigation tree, select **Environment** →**Virtual Hosts**.
2. Click the virtual host.
3. Select **Host Aliases** under Additional Properties.
4. Note each alias. An alias is composed of a host name and port number. Each alias represents a valid *host:port* combination for the Web module.

Figure 2-11 on page 71 shows the list of aliases by which *default_host* is known. The host aliases are *:80, *:9080 and *:9443. The asterisk means that any host name can be used.


Virtual Hosts > default_host > Host Aliases

A list of one or more DNS aliases by which the virtual host is known.

⊞ Preferences

Select	Host Name	Port
☐	*	9080
☐	*	80
☐	*	9443

Total 3

Figure 2-11 Default_host virtual host aliases

Compare the result with the failing URL

To compare the failing URL with the correct URL for the servlet:

1. Combine the virtual host alias, context root, and URL pattern to form the correct URL request for the servlet.
2. Compare the correct URL for the servlet with the URL that was reported in the problem.
3. If they are the same, verify that the resource file exists in the deployed application. If they are not, correct the application that is calling the resource.

In the PlantsByWebSphere ShoppingServlet example, requests for the servlet with any of the following URLs map to the default_host virtual host:

- `http://`*host*`:80/PlantsByWebSphere/servlet/ShoppingServlet`
- `http://`*host*`:9080/PlantsByWebSphere/servlet/ShoppingServlet`
- `https://`*host*`:9443/PlantsByWebSphere/servlet/ShoppingServlet`

2.3.10 Verify that the Web server plug-in is working correctly

If you have recently updated or installed the application, ensure that the Web server plug-in was regenerated and propagated to the Web server. Also, ensure that the Web server is using the new plug-in configuration file.

Check other applications on the server

Attempt to access other applications running on the same application server through the Web server. If you can access another application, this verifies that the plug-in and Web server are working with the application server, but not necessarily that the plug-in configuration is correct.

Use Default Application for access

If Default Application is installed, you can try accessing the snoop or hitcount servlets through the Web server as a quick test of the plug-in configuration.

The URLs to access these servlets are:

- http://*Web_server_host*/snoop
- http://*Web_server_host*/hitcount

Access the application directly, bypassing the plug-in

If you have regenerated the plug-in and are sure it is in use, but you still have a problem, you can bypass the Web server and access the application directly from the application server. This is not the recommended method for serving a production Web site. However, it provides a good diagnostic tool when it is not clear whether a problem is in the Web server, WebSphere Application Server, or the Web server plug-in.

To bypass the Web server plug-in and access the failing application directly through the application server Web container:

1. Find the port for the Web container in the WebSphere administrative console:

   d. Select **Servers** → **Application servers**.
   e. Click the server name.
   f. Expand **Ports** in the Communications section.
   g. Note the port number listed for *WC_defaulthost*.

2. Use the port number to access the resource from a browser. For example, if the port is 9080, the URL is: http://*host*:9080/myAppContext/myJSP.jsp.

If you can access the application through the application server but not the Web server, you are most likely experiencing a problem with the Web server plug-in.

Resolve Web server plug-in problems

These actions can help you resolve Web server plug-in problems:

1. Review the virtual host mapping for the Web modules to make sure that the virtual host mapped to the module has the alias definitions required to reach the server. (see "Verify that the virtual host configuration is correct" on page 73).

2. Review the Web module mapping to ensure that the Web module is mapped to the correct the Web server.

   To see the mapping for the modules using the administrative console, select **Applications** →**Enterprise Applications**. Click the application name to open the configuration page and click **Manage Modules**.

3. Correct any problems.
4. Regenerate the plug-in and propagate it to the Web server.
5. Restart the Web server or wait for the new configuration to be activated.

If this doesn't resolve the problem, review *WebSphere Application Server V6: Web Server Plug-in Problem Determination:*

http://www.redbooks.ibm.com/redpapers/pdfs/redp4045.pdf

2.3.11 Verify that the virtual host configuration is correct

When you install an application, you associate a virtual host with each Web module in the application. The virtual host has a set of defined host aliases, each consisting of a host name and port number. Requests that match a host alias for a virtual host are processed by servlets/JSPs in the Web module.

Web server plug-in processing

When the Web server receives a request, the plug-in checks the URI of the request against those defined in the host aliases to determine which, if any, Web module can handle the request.

If the Web server plug-in receives a request that does not match one of the virtual hosts, then the user receives an HTTP error.

Determine the virtual host for the Web module

To determine the virtual host associated with the Web module, in the administrative console, select **Enterprise Applications** → ***application_name*** → **Virtual Hosts.**

The list that results shows the virtual host mapping for each Web module in the application.

In i5/OS, you can also use the IBM Web Administration for i5/OS console to view Web module virtual host mappings by selecting **Applications** → **Manage Installed Applications**. Select the application and click **Properties**.

View and modify the virtual host definition

To view and modify a virtual host definition using the WebSphere administrative console:

1. Select **Environment** → **Virtual Hosts**.
2. Click the virtual host name.
3. Click **Host Aliases**.
4. Click the host alias name that you want to modify.
5. Edit the host name or port as needed.
6. Restart the application server.
7. Regenerate the plug-in and propagate it to the Web server.

You can also use the IBM Web Administration for i5/OS console to view and manage virtual host definitions. In the console, select **Resource Configuration** → **Manage Virtual Hosts**.

Host alias definitions

Follow these guidelines when you are verifying the host alias definitions:

- Make sure that the host aliases that are defined for a virtual host include the host name and port number of the WebSphere Application Server and the host names and port numbers that the Web server plug-in is expecting to receive from the browser.
- Mapping HTTP requests to host aliases is case sensitive and the match must be alphabetically exact. For example, the request http://www.myhost.com/myservlet does not map to any of the following:
  - http://myhost/myservlet
  - http://www.myhost.com/MyServlet
  - http://www.myhost.com:9876/myservlet
- A * wild card can be used for the host name, the port, or both. When it is used for both, any request matches this rule.

What to look for

Check the virtual host definitions to make sure that the virtual host that is associated with the Web module has a host alias defined that contains the host name and the port that matches the host name and port in the URL that is causing the failure.

If `localhost` is used as an alias entry, check the etc/hosts file to ensure that all host names (aliases) that are associated with the loop back address (127.0.0.1) are part of the same virtual host grouping (for example, *default_host*). Also, verify that no duplicate host aliases have been defined in multiple virtual hosts.

For example, in Example 2-5, the host alias `test:80` is duplicated in both virtual hosts because a URI that contains `test:80` would match aliases in both virtual hosts (`*:80` and `test:80`).

Example 2-5 *Virtual host definitions*

```
<VirtualHostGroup Name="default_host">
       <VirtualHost Name="*:80"/>
       <VirtualHost Name="*:9080"/>
</VirtualHostGroup>

<VirtualHostGroup Name="test_host">
       <VirtualHost Name="test:80"/>
       <VirtualHost Name="*:9081"/>
</VirtualHostGroup>
```

2.4 HTTP 500 errors

HTTP 500 errors indicate that an internal server problem has occurred during the process of serving the requested page. Examples of the types of problems that can cause this in WebSphere Application Server include:

- JSP processor errors
- Application code (JSP, JSF, servlet) errors
- Session errors

In this section, we show you how to identify the root cause of an HTTP 500 error and how to resolve it.

2.4.1 Collect diagnostics

Begin by collecting the following diagnostics:

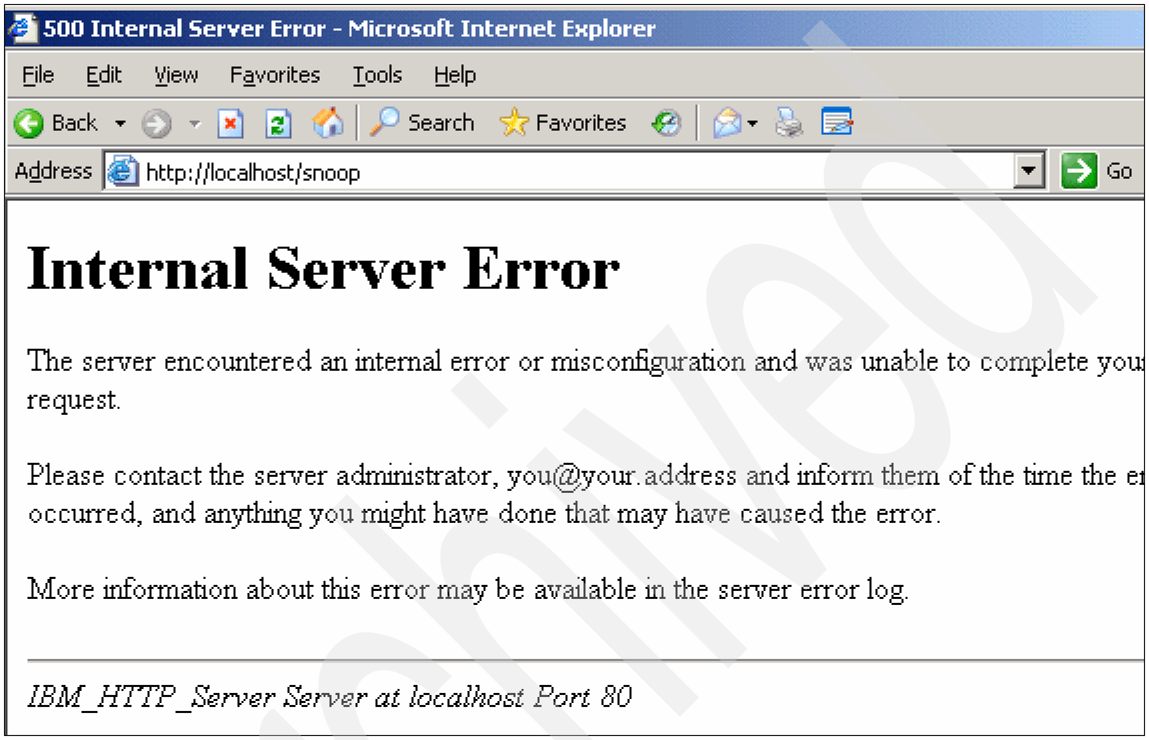
- Errors displayed by the browser, including the failing URL
- SystemOut log data for the application server (see “JVM logs” on page 97 for more information)

2.4.2 Analyze errors displayed by the browser

HTTP 500 errors often appear in the client browser with descriptive text and possibly with additional error information. The primary message is an HTTP 500 and sometimes there is additional information. However, the type of error is not always apparent to the user. Web applications often intercept and handle these types of errors, sending a generic error page to the Web client.

Internal server errors with no accompanying messages

When users receive HTTP 500 error codes and no accompanying descriptive messages (Figure 2-12), it is possible that no application servers are available to serve the file.


500 Internal Server Error - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost/snoop
Go

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete you request.

Please contact the server administrator, you@your.address and inform them of the time the e occurred, and anything you might have done that may have caused the error.

More information about this error may be available in the server error log.

IBM_HTTP_Server Server at localhost Port 80

Figure 2-12 HTTP 500 Internal Server Error

To verify that an application server is available to serve the file:

1. Determine the server or cluster that hosts the application.
2. Check the status of the application server.
3. Start the application server if it is not running.

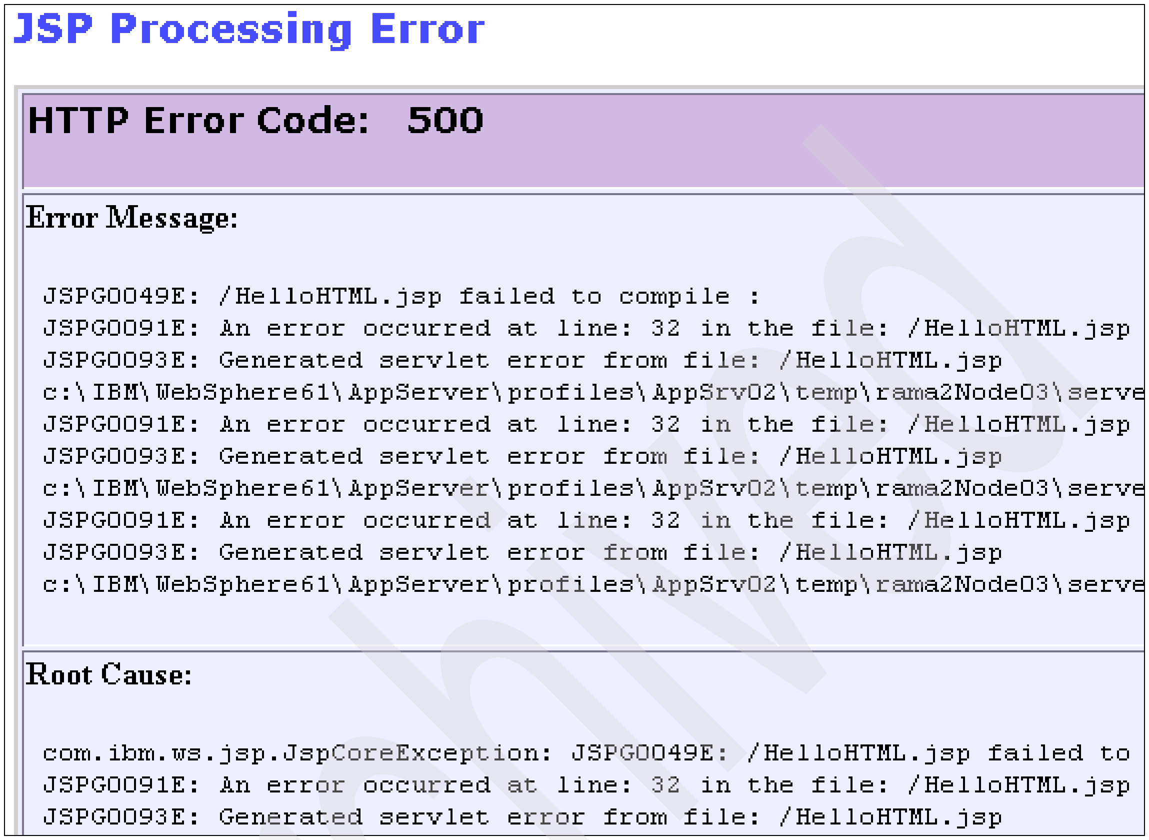
For more information, see "Managing servers and applications" on page 103.

JSP processing errors

JSP processing errors can be caused by application coding errors or a runtime error in the JSP processor in the Web container. Examples of messages for these errors include:

- JSPG0076E: Missing required attribute page...
- JSPG0049E: *jsp_name* failed to compile

Figure 2-13 shows how this might look in a Web browser.


JSP Processing Error

HTTP Error Code: 500

Error Message:

```
JSPG0049E: /HelloHTML.jsp failed to compile :
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
c:\IBM\WebSphere61\AppServer\profiles\AppSrv02\temp\rama2Node03\serve
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
c:\IBM\WebSphere61\AppServer\profiles\AppSrv02\temp\rama2Node03\serve
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
c:\IBM\WebSphere61\AppServer\profiles\AppSrv02\temp\rama2Node03\serve
```

Root Cause:

```
com.ibm.ws.jsp.JspCoreException: JSPG0049E: /HelloHTML.jsp failed to
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
```

Figure 2-13 JSP Processing Error message

Note: If you have JSPGxxxxE or JSFGxxxxE messages, see “Code error” on page 81 for more information.

IllegalStateException errors

IllegalStateException errors generally show up in a Java stack trace in the SystemOut log and can be caused by several things; the most common are invalid session objects and response generation problems.

For example, you might see “SRVE0068ESRVE0068E: could not invoke the *method_name* method on servlet *servlet_name.* Exception thrown: java.lang.IllegalStateException” with one of the following suffixes:

- Session object internals: *session_info*

- Response already committed

2.4.3 Analyze SystemOut

To analyze SystemOut, search for the following messages related to the error:

- SRVE0068E: could not invoke the *method_name* method on servlet *servlet_name.* Exception thrown: java.lang.IllegalStateException. If you have an illegal state exception error, the exception should give you an indication of the illegal state causing the problem.
- SRVE0239I and SRVE0240I, indicating that the JSP processor started normally.
- JSPGxxxxE messages, indicating a JSP error.
- JSFGxxxxE messages, indicating a JSF error.

Note the error text and the text that follows each message.

If you do not see any error messages or you see error messages other than those with the prefixes listed here, see "The next step" on page 99 for more information.

JSP processor error

Browse the SystemOut log of the server that hosts the JSP files to determine if the JSP processor has started successfully. Failure to start normally indicates a JSP processor exception. The messages in Example 2-6 indicate that the JSP processor has started normally.

Example 2-6 *JSP processor messages in SystemOut.log file*

```
WebContainer  A   SRVE0239I: Extension Factory [class
com.ibm.ws.jsp.webcontainerext.JSPExtensionFactory] was registered
successfully.
WebContainer  A   SRVE0240I: Extension Factory [class
com.ibm.ws.jsp.webcontainerext.JSPExtensionFactory] has been associated with
patterns [*.jsp *.jspx *.jsw *.jsv ].
```

If you have a JSP processor error, collect MustGather documentation and see "The next step" on page 99 for more information.

Code errors

If the JSP processor starts normally, the problem might be with the application code itself. For example, the JSP might have invalid JSP syntax that cannot be processed by the JSP processor.

Example 2-7 on page 79 shows a message that indicates a problem with JSP directive syntax.

Example 2-7 *JSP directive syntax error message*

```
Message: /test.jsp(2,1)JSPG0076E: Missing required attribute page for jsp
element jsp:include
```

Example 2-8 shows a message that indicates invalid Java syntax.

Example 2-8 *Invalid Java syntax error message*

```
com.ibm.ws.jsp.JspCoreException: JSPG0049E: /test.jsp failed to compile :
JSPG0091E: An error occurred at line: 16 in the file: /test.jsp
JSPG0093E: Generated servlet error from file: /test.jsp
```

For more information about a problem in the code, see “Code error” on page 81.

IllegalStateException: Invalid session object

When the application tries to use an invalidated session object, the IllegalStateException occurs (Example 2-9).

Example 2-9 *IllegalStateException in invalid session object*

```
[7/7/05 16:41:30:627 ART] 00000028 ServletWrappe E   SRVE0068E: Could not
invoke the service() method on servlet TestServlet. Exception thrown :
java.lang.IllegalStateException:
Session Object Internals:
id : pi55X7syi-ExTjyyhFn5Cu7
hashCode : 1449590567
create time : Thu Jul 07 16:24:54 ART 2005
last access : Thu Jul 07 16:41:30 ART 2005
max inactive interval : 1800
user name : anonymous
valid session : false
new session : true
overflowed : false
non-serializable app specific session data : {}
serializable app specific session data : {}
   at
com.ibm.ws.webcontainer.httpsession.SessionData.getValueGuts(SessionData.java(C
ompiled Code))
   at
com.ibm.ws.webcontainer.httpsession.SessionData.getValue(SessionData.java(Inlin
ed Compiled Code))
   at
com.ibm.ws.webcontainer.httpsession.SessionData.getAttribute(SessionData.java(I
nlined Compiled Code))
```

```
   at
com.ibm.ws.webcontainer.httpsession.HttpSessionFacade.getAttribute(HttpSessionF
acade.java(Compiled Code))
```

For errors with session objects, see “Invalid session object” on page 81 for more information.

IllegalStateException - Response generation problem

Examples of errors that are thrown by the HttpServletResponse interface during the response generation problem are:

- `java.lang.IllegalStateException: Response already committed`
- `java.lang.IllegalStateException: Header already sent`
- `java.lang.IllegalStateException: Cannot forward as Output Stream or Writer has already been obtained`

Example 2-10 shows the error messages generated for a “Response already committed” error.

Example 2-10 *IllegalStateException in response generation*

```
[7/8/05 20:36:25:694 ART] 0000004f ServletWrappe E   SRVE0068E: Could not
invoke the service() method on servlet TestServlet. Exception thrown :
java.lang.IllegalStateException
   at
com.ibm.ws.webcontainer.webapp.WebAppDispatcherContext.sendRedirect(WebAppDispa
tcherContext.java:486)
   at
com.ibm.ws.webcontainer.srt.SRTServletResponse.sendRedirect(SRTServletResponse.
java:810)
   at web.TestServlet.doGet(TestServlet.java:56)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)

[7/8/05 20:36:25:764 ART] 0000004f LocalTranCoor E   WLTC0017E: Resources
rolled back due to setRollbackOnly() being called.
[7/8/05 20:36:25:774 ART] 0000004f WebApp        E   SRVE0026E: [Servlet
Error]-[TestServlet]: java.lang.IllegalStateException
   at
com.ibm.ws.webcontainer.webapp.WebAppDispatcherContext.sendRedirect(WebAppDispa
tcherContext.java:486)
   at
com.ibm.ws.webcontainer.srt.SRTServletResponse.sendRedirect(SRTServletResponse.
java:810)
   at web.TestServlet.doGet(TestServlet.java:56)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
```

```
[7/8/05 20:36:25:784 ART] 0000004f SRTServletRes W   WARNING: Cannot set
status. Response already committed.
```

For response generation problems, see “Response generation errors” on page 83.

2.4.4 Root causes

Some root causes for HTTP 500 errors are:

- Code error
- Invalid session object
- Response generation errors

Code error

The root cause of this problem is usually a programming error.

For help, look up the extended error definitions

- JSPG messages:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.messages.doc/doc/JSPG.html
- JSFG messages:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.jsf.resources.messages.html

Correct the error and retry the file.

If you have not identified the problem, check to see if the problem has been documented by looking at the available online support (hints and tips, technotes, and fixes) for JSP problems at:

http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCC2GL&rankprofile=8&dc=DB520+D800+D900+DA900+DA800&dtm

If the searches fail to resolve your problem, see “The next step” on page 99.

Invalid session object

Check the servlet or JSP code that threw the exception to make sure that the session is not being invalidated too early. If that is not the case, check the session timeout interval to ensure that it is not too short.

About session objects

The session manager component uses the HttpSession interface to create a session between an HTTP client and the server. When a new session object is created, a unique session ID is assigned to it. The session ID, which is then passed as part of every request, matches the user with the session object. Session tracking is what servlets use to maintain state and user information for multiple requests.

Sessions might be invalidated automatically if there is a session timeout or they can be ended explicitly by application code. The HttpSession interface provides the following method to terminate a session explicitly:

```
public void invalidate();
```

When a session terminates, the session object and the information that is stored in it are lost permanently. The session manager unbinds any objects that are bound to the session before it destroys the session.

HttpSession interface life cycle

Figure 2-14 illustrates the HttpSession interface life cycle.

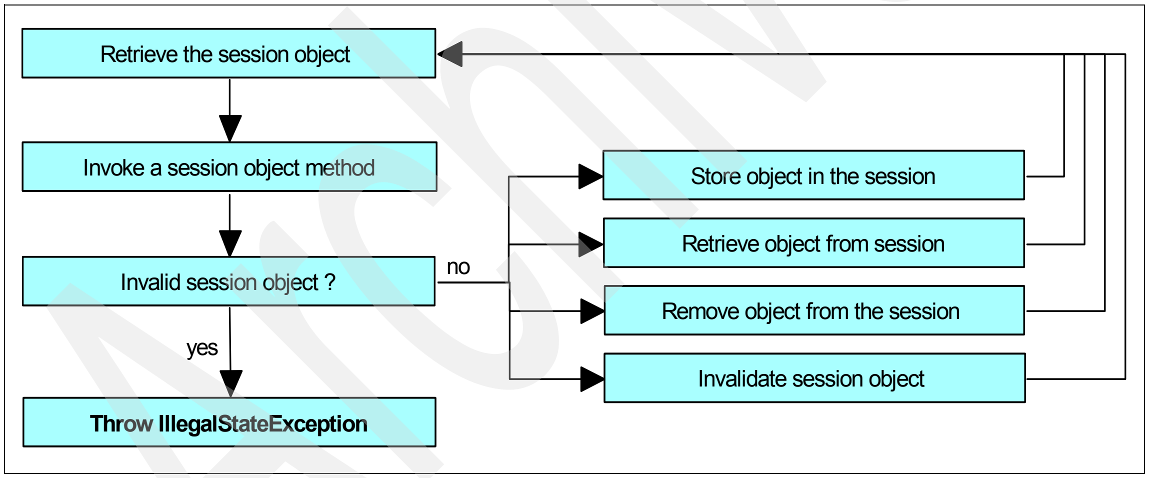

Retrieve the session object
Invoke a session object method
Invalid session object ?
no
yes
Throw IllegalStateException
Store object in the session
Retrieve object from session
Remove object from the session
Invalidate session object

Figure 2-14 HttpSession interface life cycle

Look at available online support

For more information, review the Java Servlet Specification Version 2.4, Section SRV.15.1.7 that relates to the HttpSession interface and methods definitions, to obtain more details about IllegalStateException creation causes:

http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html

For current information that is available from IBM Support about known problems and their resolution related to session management, visit:

http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCMPDS&rankprofile=8&dc=DB520+D800+D900+DA900+DA800&dtm

If these actions fail to identify your problem, see “The next step” on page 99.

Response generation errors

When an HTTP request from a client is delegated to a servlet, the service() method of the HttpServlet class is invoked. The HttpServlet class adds additional methods, such as doGet(), doPost(), doPut() and doHead(), for HTTP-based request processing.

Response already committed

The java.lang.IllegalStateException: Response already committed exception is thrown by the HttpServletResponse interface during the response generation process. If the response has been committed, you cannot execute any method that is related to HttpServletResponse object modification. For example, if you have written something in the response buffer, you cannot forward a page using the RequestDispatcher interface methods.

Other issues to look for in an application that can cause a java.lang.IllegalStateException are the following calls when the response has already been committed:

- Calling setBufferSize()
- Calling ServletResponse.reset() or ServletResponse.resetBuffer()
- Calling HttpServletResponse.sendError() or HttpServletResponse.sendRedirect().
- Calling RequestDispatcher.forward(), which includes performing a jsp:forward

Note: Remember that if you call forward() or sendRedirect() in your code, any lines of code following these still runs.

Header already sent

If you see the following message, it means that one or more headers have been committed to the client, so you cannot set that header again:

```
java.lang.IllegalStateException: Header already sent
```

Cannot forward as Output Stream, Writer already obtained

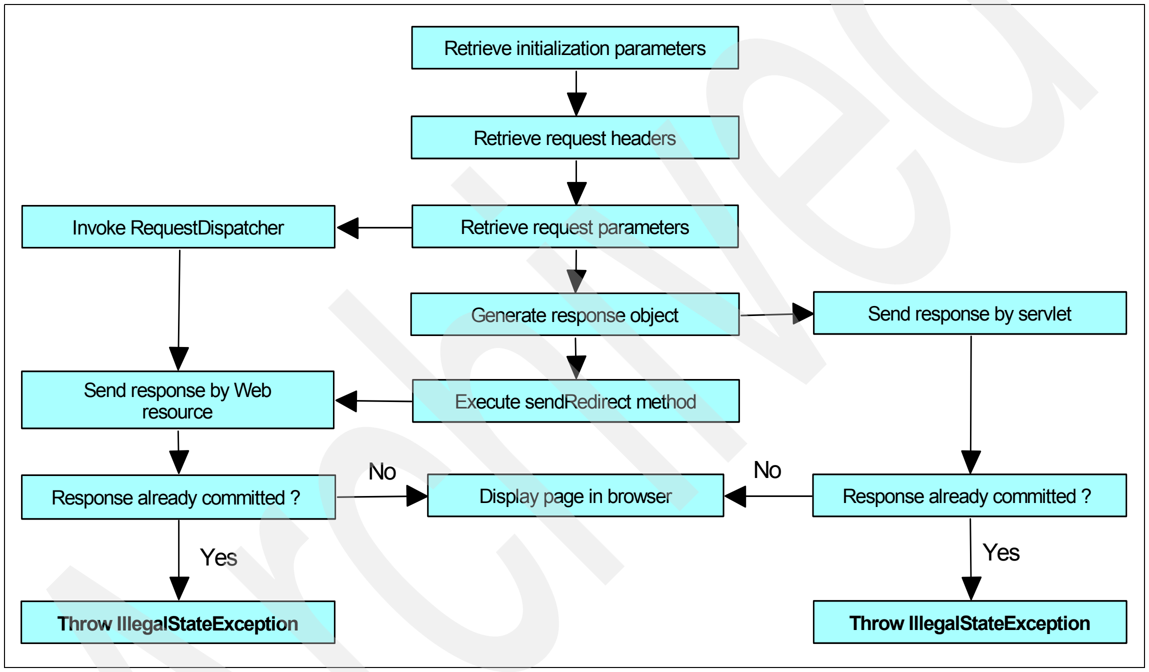
If you see the following message, it means that the calling servlet has called response.getWriter() or response.getOutputStream():

```
java.lang.IllegalStateException: Cannot forward as Output Stream or
Writer has already been obtained
```

Because the response has been written, it is unsuitable for forwarding.

The service() method life cycle

Figure 2-15 illustrates the life cycle of the service() method.



Retrieve initialization parameters
Retrieve request headers
Invoke RequestDispatcher
Retrieve request parameters
Generate response object
Send response by servlet
Send response by Web resource
Execute sendRedirect method
No
No
Response already committed ?
Display page in browser
Response already committed ?
Yes
Yes
Throw IllegalStateException
Throw IllegalStateException

Figure 2-15 Servlet service() method life cycle

Look at available online support

If you still have not identified the cause of the problem, see the Java Servlet Specification Version 2.4 at the following URL to obtain more details about the causes of IllegalStateException generation in the response generation process:

http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html

Review the following sections:

- SRV.14.2.5 RequestDispatcher interface
- SRV.14.2.16 ServletRequest interface

- SRV.14.2.22 ServletResponse interface

For current information from IBM Support about known issues and resolutions that relate to session management, go to the following URL:

```
http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCMPDF&rank
profile=8&dc=DB520+D800+D900+DA900+DA800&dtm
```

If these steps do not resolve your problem, see "The next step" on page 99 for more information.

2.5 Incorrect information displayed on Web pages

If users are not receiving HTTP 404 or 500 errors, but are seeing incorrect page results in the Web browser, you could be experiencing a Web container problem. These problems are often a result of improper settings for the application in the deployment descriptors, application packaging, or Web container settings in the runtime environment.

The following symptoms can indicate a problem with the Web container:

- Users report that pages appear with missing elements.

  If a Web page appears but is missing static resources such as text, images, or file segments, see "Static resources not displayed" on page 86.
- Users report seeing an old version of application pages.

  If a JSP file has been modified and deployed to the server but the changes do not appear in the browser interface, make sure that the application has been enabled for JSP reloading. See "Web resources not reloading" on page 88.
- Users report that pages contain invalid information such as multiple question marks or garbage characters or that input is not interpreted correctly.

  It is possible that the application uses Double Byte Character Set (DBCS) characters (Japanese, Chinese, Korean languages) or specific characters for other languages that are not included in the default character encoding. In these cases, a correct character encoding configuration is necessary to display and process this information without problems. For more details related to encoding configurations, go to "Encoding and internationalization issues" on page 91.
- Users report lost data during a session.

  Users repeatedly have to enter information that should be saved during the session, lose shopping cart information, or have other short-term data loss.

This data loss might be caused by a problem with HTTP sessions. For information about troubleshooting HTTP session problems, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/rtrb_httpsessprobs.html

2.5.1 Static resources not displayed

If the browser displays text output that is related to a JSP or servlet Web page but not images or HTML files, you could have a problem in the Web module packaging or with how the files are referenced in the application. Another possibility is that the file serving feature needs to be turned on for your application.

Verify the static resource file locations

Verify that your files are in the right place and that the document root directory of the Web application module follows the J2EE standard (the document root is in the *web_module*.war directory of the deployed application EAR file). Typically this directory is in this location:

app_server_root/profiles/*profile*/installedApps/*node*/*application*.ear/*web_module*.war/

If the image files are in a subdirectory of the document root, verify that the reference to the image reflects that (Example 2-11).

Example 2-11 Image reference in HTML tags

```
File Location: <web_module_name.war>/images/test.gif
Correct HTML tag: <img SRC="images/test.gif">
Incorrect HTML tag: <img SRC="test.gif">
```

Note: Do not place files to be served to the client in the WEB-INF directory.

Check the file serving feature

File serving is how a Web application serves static file types (HTML, images, and style sheets). This process uses the enable file servlet, also known as the file serving servlet or file serving enabler. This servlet serves up any resource file that is packaged in the WAR file, and the file serving attribute is set to **true** by default.

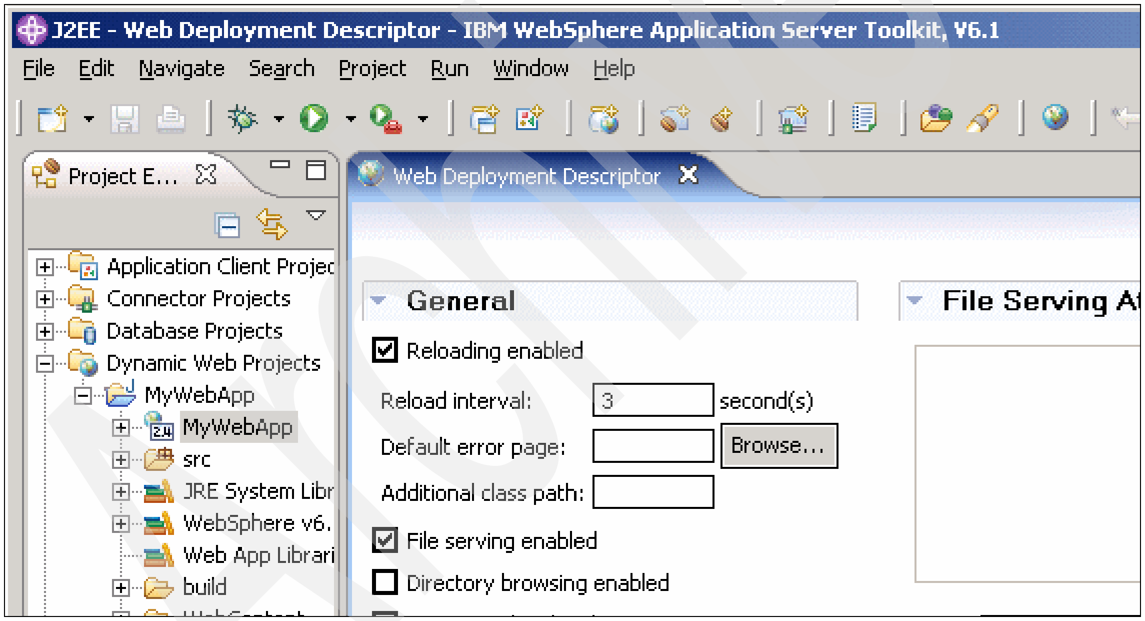
File serving is implemented by setting the fileServingEnabled property to **true** when you configure the Web module. If it is set to **false**, the Web server plug-in does not send requests for static content to the application server but leaves it up to the Web server to serve them. Serving the page from the Web server provides a shorter path to the page and usually provides more customization options than the file servlet can offer.

The fileServingEnabled property is in the ibm-web-ext.xmi configuration file at:

profile_root/installedApps/*node*/*application*.ear*web_module*.war/WEB-INF/ibm-web-ext.xmi

To update the property using the Application Server Toolkit:

1. Go to the Project Explorer view in the J2EE perspective and select the target Web application module.
2. Double-click the Web deployment descriptor and click the Extensions tab to see the IBM Web module extensions.
3. In the General section, select **File serving enabled** (Figure 2-16) to enable the static file serving or clear it to disable the static file serving.


J2EE - Web Deployment Descriptor - IBM WebSphere Application Server Toolkit, V6.1
File Edit Navigate Search Project Run Window Help
Project E...
Application Client Projec
Connector Projects
Database Projects
Dynamic Web Projects
MyWebApp
MyWebApp
src
JRE System Libr
WebSphere v6.
Web App Librari
build
Web Deployment Descriptor
General
File Serving At
Browse...

☑ Reloading enabled
Reload interval: 3 second(s)
Default error page:
Additional class path:
☑ File serving enabled
☐ Directory browsing enabled

Figure 2-16 Enabling file serving

4. Save the Web deployment descriptor file.
5. Redeploy the Web module.
6. Regenerate the Web server plug-in and propagate it to the Web server.
7. Stop and restart the Web server or allow enough time for the new plug-in configuration to be reloaded.

8. Retry the Web request.

For more information about this feature, see:

- *File serving*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/cweb_filserv.html

- *Customizing SimpleFileServlet* to disable file serving at:

  http://www.ibm.com/support/docview.wss?uid=swg21116838

- Supported assembly tools in WebSphere Application Server V6:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/catk_assemblytools.html

2.5.2 Web resources not reloading

If, after modifying and saving a servlet or JSP file, the change does not show up in the browser, you need to check the reload settings in the Web module configuration and the JSP runtime reload settings.

Web module reloading

For Web resources such as servlets and JSPs, the Web container reloads a Web module only when the IBM extension reloadingEnabled in the ibm-web-ext.xmi file is set to **true**. You can do this when you are editing the extended deployment descriptors of the Web module in a development or assembly tool.

To configure these settings with the Application Server Toolkit:

1. From the Project Explorer view of the J2EE perspective, double-click the Web deployment descriptor and click the Extensions tab.
2. In the General section, select **Reloading enabled** (see Figure 2-16 on page 87). If it is already selected, set the **Reload interval** lower.
3. Save the Web deployment descriptor file.
4. Redeploy the Web module.
5. Regenerate the Web server plug-in and propagate it to the Web server.
6. Stop and restart the Web server or wait for the new plug-in file to take effect.
7. Retry the Web request.

The entries in ibm-web-ext.xmi look similar to those in Example 2-12.

Example 2-12 Web module reloading settings in ibm-web-ext.xmi

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<webappext:WebAppExtension xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:webappext="webappext.xmi" xmi:id="WebAppExtension_1176282340062"
reloadInterval="3" reloadingEnabled="true" additionalClassPath=""
fileServingEnabled="true" directoryBrowsingEnabled="false"
serveServletsByClassnameEnabled="true">
  <webApp href="WEB-INF/web.xml#WebApp_ID"/>
</webappext:WebAppExtension>
```

Configure JSP runtime reloading

JSP files can be translated and compiled at runtime when the JSP file or its dependencies are modified. This is known as JSP reloading, and it is enabled through the reloadEnabled JSP engine parameter in the ibm-web-ext.xmi configuration file.

To configure JSP reload using the Application Server Toolkit:

1. From the Project Explorer view of the J2EE perspective, double-click the Web deployment descriptor and select the Extensions tab.
2. In the JSP Attributes section (Figure 2-17), click **Add**.

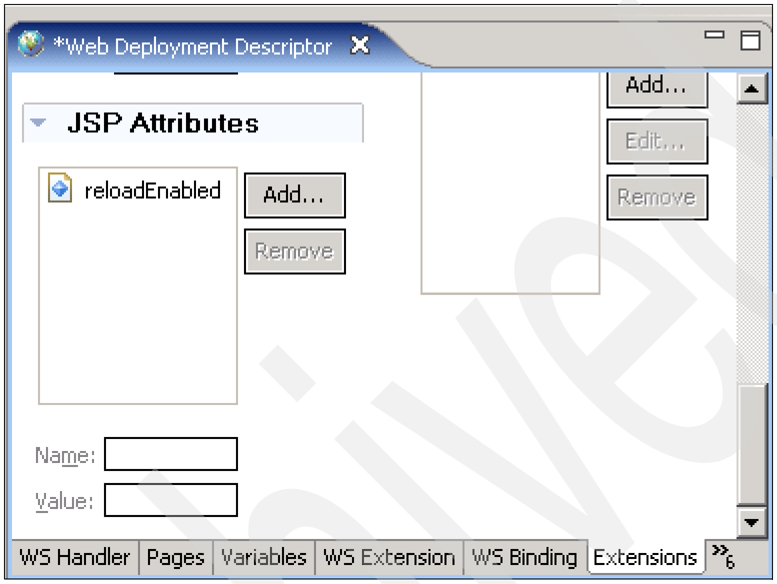


Figure 2-17 JSP attributes in IBM Web module extensions

3. Enter `reloadEnabled` in the Name field, `true` in the Value field, and click **Finish**.
4. Save the Web deployment descriptor file.
5. Redeploy the Web module.
6. Regenerate the Web server plug-in and propagate it to the Web server.
7. Stop and restart the Web server.
8. Retry the Web request.

The JSP engine settings are stored in ibm-web.ext.xmi (Example 2-13).

Example 2-13 JSP engine settings in ibm-web-ext.xmi

```
?xml version="1.0" encoding="UTF-8"?>
<webappext:WebAppExtension xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:webappext="webappext.xmi" xmi:id="WebAppExtension_1176282340062"
reloadInterval="3" reloadingEnabled="true" additionalClassPath=""
```

```
fileServingEnabled="true" directoryBrowsingEnabled="false"
serveServletsByClassnameEnabled="true">
  <webApp href="WEB-INF/web.xml#WebApp_ID"/>
  <jspAttributes xmi:id="JSPAttribute_1176283583578" name="reloadEnabled"
value="true"/>
</webappext:WebAppExtension>
```

For more available JSP attributes and details about the reload processing sequence, see “JavaServer™ Pages™ (JSP) runtime reloading settings” in the WebSphere Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rweb_jspreloading.html

2.5.3 Encoding and internationalization issues

Encoding and internationalization problems usually appear as garbage characters in Web pages. Another common symptom is that input from the user is interpreted incorrectly.

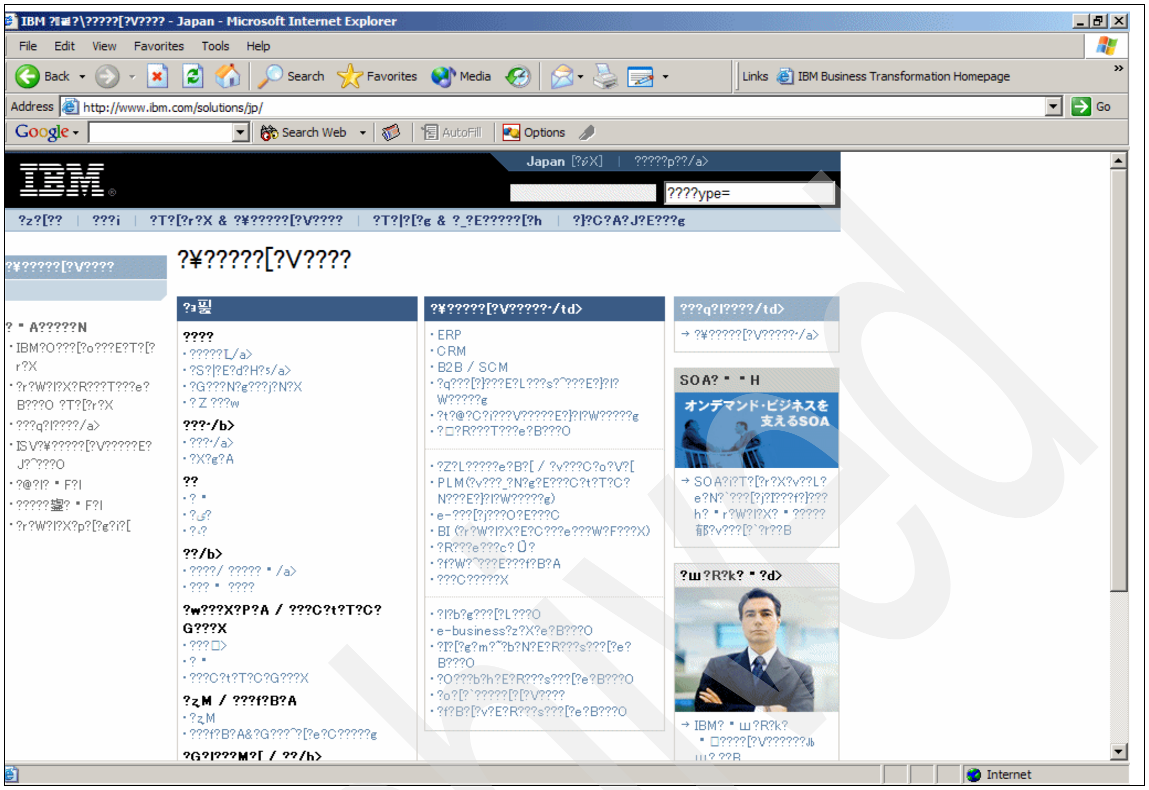
DBCS character processing is a common issue for encoding problems. DBCS is used for languages such as Chinese, Korean, and Japanese, where a single byte is not sufficient to represent all characters in the alphabet. Proper coding and configuration usually resolves these problems.

Character encoding

Web components usually use the java.io.PrintWriter object to produce responses; PrintWriter automatically encodes with ISO 8859-1 character encoding. Servlets can also output binary data using java.io.OutputStream objects, which perform no encoding.

An application that cannot use the default encoding must explicitly set a different encoding.

Figure 2-18 on page 92 shows a Web page with improper encoding settings.


IBM ?ｲ■?\?????[?V???? - Japan - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://www.ibm.com/solutions/jp/

Japan [?ｶX] | ?????p??/a>

????ype=

?z?[?? | ???i | ?T?[?r?X & ?¥?????[?V???? | ?T?|?[?g & ?_E?????[?h | ?]?C?A?J?E???g

?¥?????[?V????

?¥?????[?V????

? ･ A?????N
・IBM?O???[?o???E?T?[?r?X
・?r?W?l?X?R???T???e?B???O ?T?[?r?X
・???q?l????/a>
・ISV?¥?????[?V?????E?J?`???O
・?@?l? ･ F?l
・?????塁? ･ F?l
・?r?W?l?X?p?[?g?i?[

?ｮ짎

????
・?????L/a>
・?S?|?E?d?H?s/a>
・?G???N?g???j?N?X
・?Z ???w

???･/b>
・???･/a>
・?X?g?A

??
・? ･
・?ｮ?
・?･?

??/b>
・????/ ????? ･ /a>
・??? ･ ????

?w???X?P?A / ???C?t?T?C?G???X
・???□>
・? ･
・???C?t?T?C?G???X

?ｚM / ???f?B?A
・?ｚM
・???f?B?A&?G???`?[?e?C?????g

?G?I???M?[/ ??/b>

?¥?????[?V?????･/td>
・ERP
・CRM
・B2B / SCM
・?q???[?]???E?L???s?`???E?]?I?W?????g
・?t?@?C?i???V?????E?]?I?W?????g
・?□?R???T???e?B???O

・?Z?L?????e?B?[/ ?v???C?o?V?[
・PLM(?v???_N?g?E???C?t?T?C?N???E?]?I?W?????g)
・e-???[?j???O?E???C
・BI (?r?W?l?X?E?C???e???W?F???X)
・?R???e???c? Ü?
・?f?W?`???E???f?B?A
・???C?????X

・?I?b?g???[?L???O
・e-business?z?X?e?B???O
・?I?[?g?m?`?b?N?E?R???s???[?e?B???O
・?O???b?h?E?R???s???[?e?B???O
・?o?[?`?????[?[?V????
・?f?B?[?v?E?R???s???[?e?B???O

???q?l????/td>
→ ?¥?????[?V?????･/a>

SOA? ･ ･ H

オンデマンド・ビジネスを支えるSOA

→ SOA?i?T?[?r?X?v??L?e?N?`???[?j?I???f?]???h? ･ r?W?l?X? ･ ?????郁?v???[?`?r??B

?ш?R?k? ･ ?d>

→ IBM? ･ ш?R?k?
･ □????[?V??????лш?.??B

Internet

Figure 2-18 Web page with invalid character encoding settings

For Web components, three encodings must be considered:

- Request
- JSP
- Response

Request encoding

Request encoding is the character encoding where parameters in an incoming request are interpreted. Currently, many browsers do not send a request encoding qualifier with the content-type HTTP header. In such cases, a Web container uses the default encoding, which is ISO-8859-1, to parse request data.

If the client has not set character encoding and the request data is encoded with a different encoding from the default, the data is not interpreted correctly.

To correct this situation, you can use the setCharacterEncoding(String enc) method to override the character encoding that is supplied by the container (Example 2-14 on page 93).

Example 2-14 setCharacterEncoding() method implementation

```
public class TestServlet extends HttpServlet {

   public void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException {

         req.setCharacterEncoding("UTF-8");
         String name = req.getParameter("name");
         resp.setContentType("text/html;charset=UTF-8");
         PrintWriter out = resp.getWriter();
         out.println("<html><body><h1>");
         out.println("Your name is "+name);
         out.println("</h1></body></html>");
   }
}
```

You must call the method before parsing any request parameters or reading any input from the request. Calling the method or tag after data has been read does not affect the encoding.

Page encoding

For JSP, page encoding is the character encoding where the file is encoded. For JSP in standard syntax, it is determined from the following sources:

- The pageEncoding attribute of the page directive of the page
- The charset value of the contentType attribute of the page directive

If none of these is provided, ISO-8859-1 is used as the default page encoding (Example 2-15). The pageEncoding and contentType attributes determine the page character encoding only for the file that physically contains the page directive.

Example 2-15 Page directive implementation

```
<HTML>
<HEAD>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="ISO-8859-15" %>

<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>Receiving parameters</TITLE>
</HEAD>
<BODY>
<H1>
Your data is <%= request.getParameter("name") %>
</H1>
</BODY>
```

```
</HTML>
```

Response encoding

Response encoding is the character encoding of the textual response that is generated from a Web component. A Web container sets an initial response encoding for a JSP page from the following sources:

- The charset value of the contentType attribute of the page directive
- The encoding specified by the pageEncoding attribute of the page directive

If none of these is provided, ISO-8859-1 is used as the default response encoding.

The setContentType() method in a servlet can be called to change the character encoding. Calls made after the getWriter() method has been called or after the response is committed do not affect the character encoding.

Struts character encoding settings

For Struts, consider the following methods for encoding:

- Request character encoding

  In a Struts environment, call the setCharacterEncoding() method in the ActionForm (Example 2-16).

Example 2-16 Request encoding in Struts implementation

```
public class PostMessageForm extends ActionForm {
   public void reset(ActionMapping mapping, HttpServletRequest request) {
      try {
         request.setCharacterEncoding("UTF-8");
      }catch (UnsupportedEncodingException e) {
         e.printStackTrace();
      }
   }
}
```

- Response character encoding

  Specify the response encoding using the contentType attribute with a charset value in the JSP page directive.

WebSphere Application Server settings

In normal circumstances, programmers are expected to determine the encoding settings in the application. However, when you are diagnosing problems, it is useful to set them temporarily or override them in the runtime configuration.

Request/response character encoding

WebSphere determines the character encoding that is used for request/response by parsing the client input values in getParameter() and writing the output based on the value in the accept-language header of the incoming request.

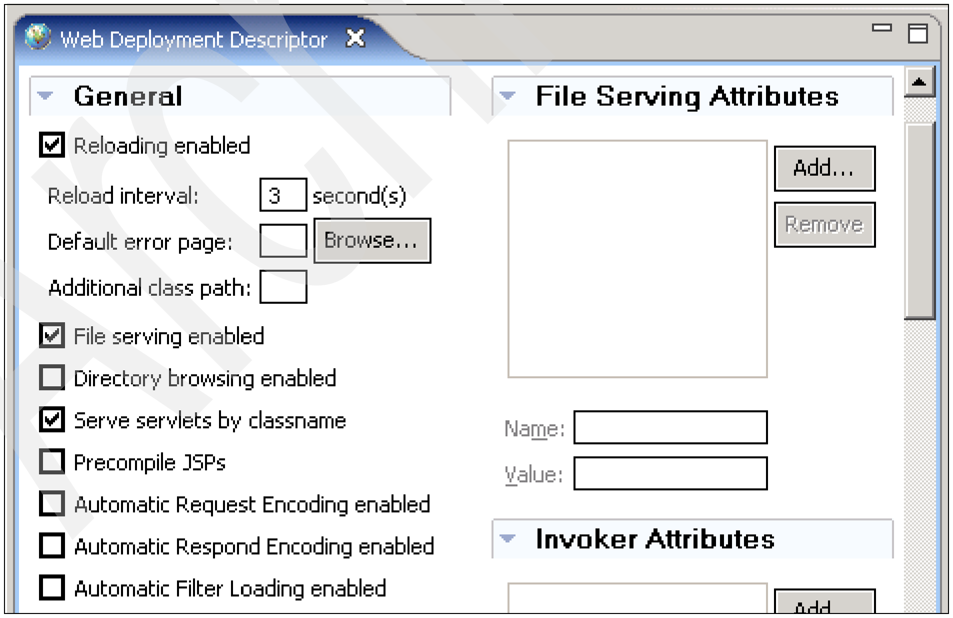
The language value and corresponding character encoding names are associated in the encoding.properties file in the *app_server_root*/properties directory. You can override the definition in this file by setting the client.encoding.override JVM command line argument.

For information about using this argument, see “Configuring application servers for UCS Transformation Format” at:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/
com.ibm.websphere.base.doc/info/aes/ae/trun_svr_utf.html
```

autoRequestEncoding and autoResponseEncoding

Programmers set request and response encodings and response content types using available methods in the Servlet specification. However, you can specify the autoRequestEncoding and autoResponseEncoding extensions so that the application server sets the encoding values and content type, using the Application Server Toolkit. The settings are found in the Web Deployment Descriptor on the Extensions tab (Figure 2-19).


Web Deployment Descriptor

General

- [✓] Reloading enabled
- Reload interval: 3 second(s)
- Default error page:
- Additional class path:
- [✓] File serving enabled
- [ ] Directory browsing enabled
- [✓] Serve servlets by classname
- [ ] Precompile JSPs
- [ ] Automatic Request Encoding enabled
- [ ] Automatic Respond Encoding enabled
- [ ] Automatic Filter Loading enabled

File Serving Attributes

Name:
Value:

Invoker Attributes

Figure 2-19 Automatic request/response encoding settings

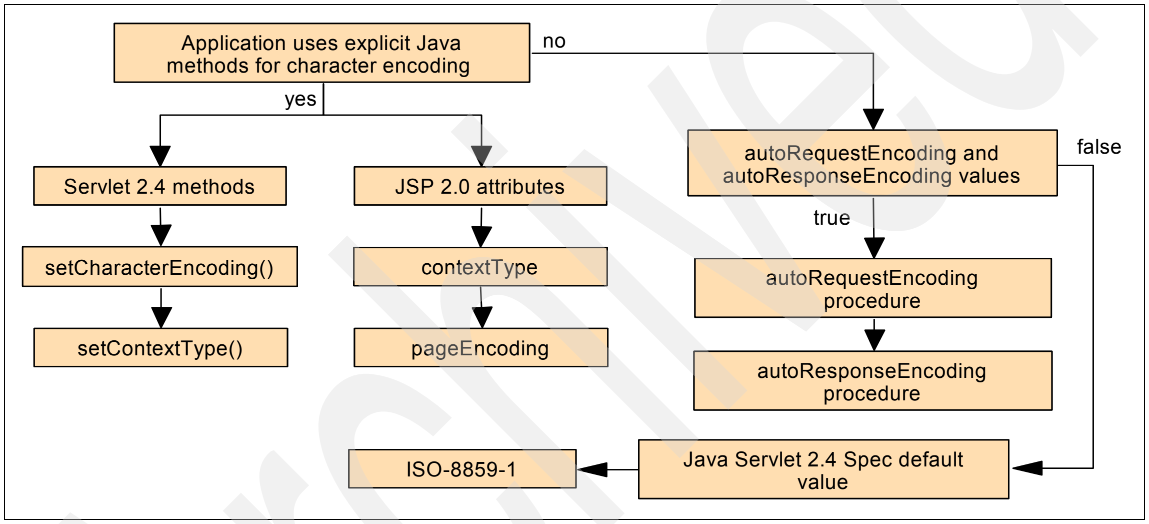
The default value for both is false, which means that the request and response character encoding is set to ISO-8859-1, the Servlet 2.4 Specification default.

For a description of Web container behavior when these values are set to **true**, see "autoRequestEncoding and autoResponseEncoding," an article in the WebSphere Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cweb_autoreq.html

Summary

Figure 2-20 shows a summary of how the encoding values are determined.



Application uses explicit Java methods for character encoding
no
yes
Servlet 2.4 methods
setCharacterEncoding()
setContextType()
JSP 2.0 attributes
contextType
pageEncoding
autoRequestEncoding and autoResponseEncoding values
false
true
autoRequestEncoding procedure
autoResponseEncoding procedure
Java Servlet 2.4 Spec default value
ISO-8859-1

Figure 2-20 Encoding determination process

Look at available online resources and support

If none of these steps fixes your problem, the following resources might be helpful:

- *Globalize your On Demand Business* for tips on character encoding:

  http://www-306.ibm.com/software/globalization/j2ee/encoding.jsp

- *JSR-000154 Java Servlet 2.4 Specification* for details about character encoding methods:

  http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html

- *JSR-000152 JavaServer Pages 2.0 Specification*, for issues related to internationalization:

  http://jcp.org/aboutJava/communityprocess/final/jsr152/index.html

- *Internationalization: Resources for learning:*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rin_resources.html

For current information available from IBM Support about known issues and resolutions that are related to encoding, see:

http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCVRZX&rankprofile=8&dc=DB520+D800+D900+DA900+DA800&dtm

If these steps still do not resolve your problem, see "The next step" on page 99..

2.6 Collect diagnostic data

This section provides guidance for collecting log files and trace data that are useful in diagnosing Web container problems.

2.6.1 JVM logs

JVM logs, often referred to as SystemOut and SystemErr logs, are created for every WebSphere Application Server process (application server, cluster member, node agent, and deployment manager). They can be found in the following locations:

- WebSphere Application Server V6.x for IBM z/OS

  The JVM logs are located in the address space output. Usually a section labeled SYSOUT has diagnostic data from the JVM that runs in the servant region.

- WebSphere Application Server V6.x (distributed and i5/OS)

  The JVM log files are by default named SystemOut.log and SystemErr.log. The default location for the SystemOut and SystemErr logs is:

  – *profile_root*/logs/*server_name*/SystemOut.log
  – *profile_root*/logs/*server_name*/SystemErr.log:

The location of application server logs is configurable.

1. Select **Troubleshooting** → **Logs and Trace** in the navigation bar.
2. Click the server name.
3. Select **JVM logs**.

2.6.2 IBM HTTP Server, HTTP Server (powered by Apache) log files

These Web servers log every client request in the access.log file. If client users experience errors, such as an error logging on to a content server, or errors importing a document, you can use the access.log file to verify the resource manager Web address and to verify that the client request is getting through to the Web server.

By default, two logs in *Web_server_root/*logs are used:

- access.log
- error.log

You can use the ErrorLog and CustomLog directives to configure the location of these logs. You can also use the LogFormat directive to configure the format and combine all the logs into one log. For information about using these directives, see the comments in the httpd.conf file for the Web server.

2.6.3 Trace Web server requests to WebSphere (i5/OS)

When an HTTP 404 error occurs in the browser, the problem is often caused by a problem with the plug-in. The plug-in file has the rules that the HTTP server follows to determine what content it can pass to the WebSphere Application Server. When HTTP is unable to match any of these rules in the plug-in file, it tries to serve the page based on its own configuration and the HTTP server sends the 404 message.

Collect the TRCTCPAPP trace

To see whether the HTTP Apache server is passing requests properly to WebSphere, trace the HTTP as follows:

1. On the i5/OS command line, issue the following command to enable the trace (replace *server_name* with the name of your HTTP server instance):

   ```
   TRCTCPAPP APP(*HTTP) HTTPSVR(server_name) TRCLVL(*VERBOSE)
   ```

2. Reproduce the error (make sure that the browser cache has been cleared so that it makes a fresh request to the server).
3. To turn off the trace, issue the following command:

   ```
   TRCTCPAPP APP(*HTTP) SET(*OFF)
   ```

   This will produce a spool file with a file name of QZSRHTTPTR in the out queue of the user profile that issued the trace commands.

2.7 The next step

The symptoms and problem areas in this paper are those that you are more likely to experience. However, there are other things that can go wrong, or the cause of the problem might be related to a component other than the Web container. This section provides information about obtaining additional support.

2.7.1 Search online support

For hints and tips, technotes, and fixes about Web container problems, see:

`http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCMPDF`

If the problem is performance related, these references are useful for configuring the JSP engine for optimal performance:

- *JSP engine configuration parameters*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rweb_jspengine.html`

- *JSP class file generation*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cweb_jspclassfiles.html`

- *JSP run time compilation settings* (to disable run time compilation)

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rweb_jspdis.html`

- *Packages and directories for generated .java and .class files*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cweb_javapkg.html`

Note: These URLs are for WebSphere Application Server Network Deployment topics on distributed platforms. If you are using a different package or platform, you can find the comparable item by searching the title in the Information Center.

2.7.2 Reevaluate the symptoms

If, after going through this process, you still have an undiagnosed problem, see *Approach to Problem Determination in WebSphere Application Server V6* at:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf`

Review the problem classifications to see if there are any other components that might be causing the problem.

2.7.3 Contact IBM

If your search does not provide any information that is relevant to your problem and you feel sure that you have a Web container problem, it might be time to contact IBM support.

Select the MustGather document from the following list that most closely resembles your problem. If you are not sure which that is, use the first MustGather listed (Web container errors).

The MustGather documents for Web container problems are:

- *MustGather: Web container errors on WebSphere Application Server V6.1*

  `http://www-1.ibm.com/support/docview.wss?uid=swg21252138`
- *MustGather: JavaServer Pages (JSP) problems*

  `http://www-1.ibm.com/support/docview.wss?uid=swg21255205`
- *MustGather: Java Server Faces (JSF) problems in V6.x*

  `http://www.ibm.com/support/docview.wss?uid=swg21198110`
- *MustGather: i18n (Internationalization) / Double Byte Character Set (DBCS)*

  `http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21141732`

Be sure to note all of the diagnostic work that you have done to minimize the amount of time that it takes IBM Support to assist you.

2.8 Approaching problems with Web servers and plug-ins

WebSphere Application Server supplies Web server plug-ins for a range of Web server types. You can find the supported Web server products in “System Requirements for WebSphere Application Server V6.1” at:

`http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007651`

If you think you have a problem with a Web server or plug-in:

- Make sure that you are using a supported Web server.
- Make sure that the Web server is responding.
- Make sure that the plug-in configuration file is current.

This section gives you a general approach to problem determination for the Web servers and Web server plug-ins that are supplied with WebSphere Application Server or IBM operating systems. For other Web server types, see the documentation for that specific product.

2.8.1 Web servers

When an application server does not respond to incoming requests, check to see if static HTML pages can be served by the HTTP Server. If not, the problem is probably caused by the HTTP Server. The HTTP Server process might have stopped unexpectedly (a crash) or it might be running but not responding to requests (a hang). You can check to see if the Apache process (httpd) is running on your operating system to determine if the process has crashed or if it is hanging. Other possible problems with HTTP Server include configuration issues, caching issues, authentication problems, and SSL problems. If the HTTP Server is able to serve static pages, the root cause of the problem is probably with the Web server plug-in or the application server itself.

Note: There is a possibility that the HTTP Server could fail to start due to a Web server plug-in problem. For more information about this, see *WebSphere Application Server V6: Web Server Plug-in Problem Determination* at:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp4045.pdf`

IBM HTTP Server

IBM HTTP Server V6.1 is included with WebSphere Application Server V6.1, and you use the administrative console to administer it. Other Web servers can be configured with WebSphere Application Server. In general, the same problems scenarios occur with them.

You can find problem determination strategies for IBM HTTP Server issues in the following resources:

- IBM HTTP Server Support

  `http://www-306.ibm.com/software/webservers/httpservers/support`

- IBM HTTP Server, Version 6.1 Information Center

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.ihs.doc/info/welcome_ihs.html`

- IBM HTTP Server, Version 6.1 Information Center, *Troubleshooting IBM HTTP Server*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs`

- *MustGather: Read first for IBM HTTP Server*

  http://www-1.ibm.com/support/docview.wss?rs=177&uid=swg21192683

HTTP Server (powered by Apache)

HTTP Server (powered by Apache) is included with the i5/OS licensed programs and must be selected for installation. You can find problem determination strategies for HTTP Server (powered by Apache) issues in the following resources:

- IBM HTTP Server for iSeries Support

  http://www-03.ibm.com/servers/eserver/iseries/software/http/services/service.html

- IBM HTTP Server for iSeries documentation

  http://www-03.ibm.com/servers/eserver/iseries/software/http/docs/doc.html

2.8.2 Web server plug-in

The Web server plug-in enables the Web server to send requests for dynamic content to Web applications (servlets and JSPs) that are installed in WebSphere Application Server. The configuration file used by the plug-in is created in the WebSphere Application Server system and propagated to the Web server.

To determine if you have a plug-in problem, first ensure that the Web server is responding. Then, try to access a servlet or JSP directly on the application server, bypassing the plug-in. You can do this by using the HTTP transport port (port 9080 by default) for the application server in the URL to access the servlet or JSP. For example, you can try to access the snoop servlet, which is one of the WebSphere Application Server samples:

```
http://localhost:9080/snoop
```

If the servlet or JSP can be accessed through the HTTP transport but not through the Web server plug-in, you are likely experiencing a problem with the plug-in. On the other hand, if it cannot be accessed through the HTTP transport or the plug-in, it is probably an application server problem. Potential Web server plug-in problems include configuration issues (especially with the plugin-cfg.xml configuration file) and failures to send requests to an application server.

WebSphere system messages for the plug-in begin with PLGN, PLGC, and PLPR.

Note: For problem determination strategies for the Web server plug-in, see *WebSphere Application Server V6: Web Server Plug-in Problem Determination* at:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp4045.pdf`

2.9 Managing servers and applications

This section includes information about managing applications and application servers in WebSphere Application Server.

2.9.1 Check the status of an application

To see the status of each module on the server, select **Applications** → **Enterprise Applications.**

This action displays a table of the applications that are deployed in the cell. The status icon in the Application Status field indicates the status of the application. Moving your mouse over the icon displays the status. Any status other than Started indicates a problem. However, an Unavailable status might simply mean that the node agent for the server that the application is running on is down.

2.9.2 Determine the server or cluster that hosts the application

To view the mapping of the application modules to servers, select **Enterprise Applications** → ***application_name*** → **Manage Modules.**

The list that results from this action shows the server mapping for each module in the application.

2.9.3 Check the status of application modules on the target servers

To see the status of each module on the server it is mapped to:

1. Select **Enterprise Applications** → ***application_name*** → **Manage Modules.**
2. Click the module and select **Target specific application status.**

2.9.4 Check the status of application servers

You can check the status of an application server using the administrative console or by command.

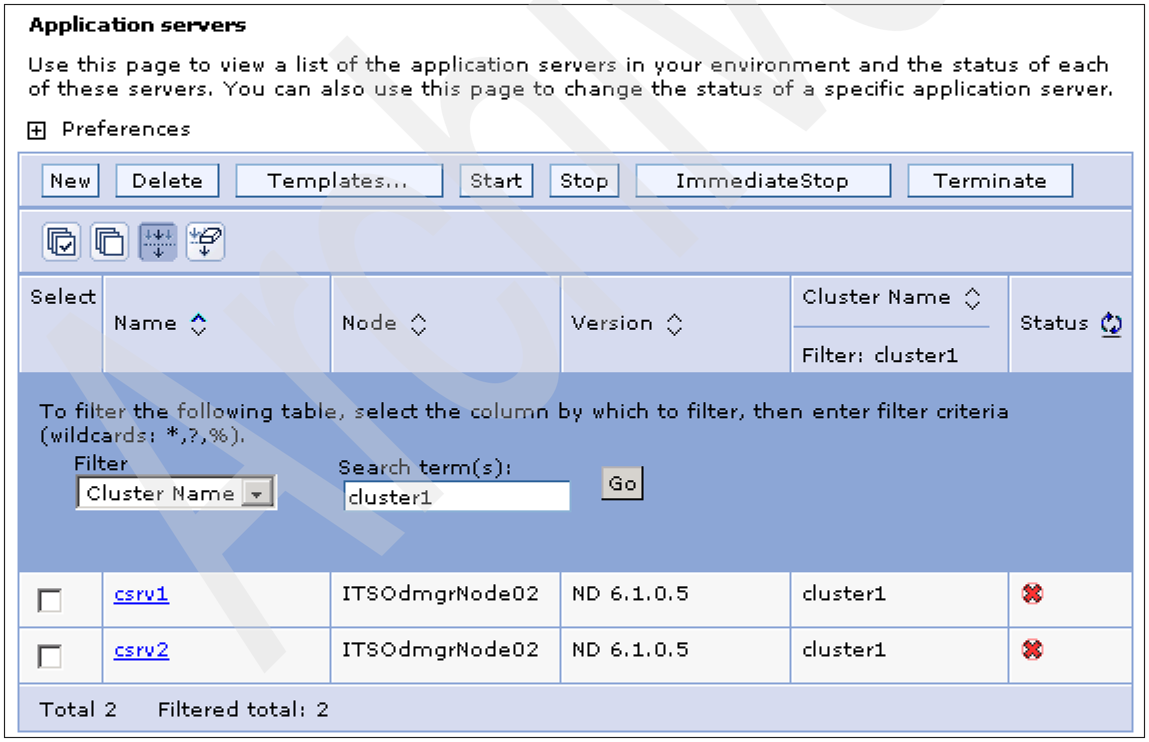
Note: In a single server environment, the administrative process runs on the application server. If you can access the administrative console, the application server is active.

Using the administrative console

To use the administrative console:

1. Select **Servers** →**Application Servers** to list the servers (Figure 2-21).

Tip: Use the filter function to manage how the servers are listed. For example, you can filter on a cluster name to see all the servers in a cluster.



Application servers

Use this page to view a list of the application servers in your environment and the status of each of these servers. You can also use this page to change the status of a specific application server.

⊞ Preferences

To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *,?,%).

Filter: Cluster Name — Search term(s): cluster1 — Go

Select	Name	Node	Version	Cluster Name (Filter: cluster1)	Status
☐	csrv1	ITSOdmgrNode02	ND 6.1.0.5	cluster1	✗
☐	csrv2	ITSOdmgrNode02	ND 6.1.0.5	cluster1	✗

Total 2 Filtered total: 2

Figure 2-21 Display of servers in a cluster

2. The last column to the right has an icon that indicates the status of each server. Figure 2-22 on page 105 shows the icons and the corresponding status.

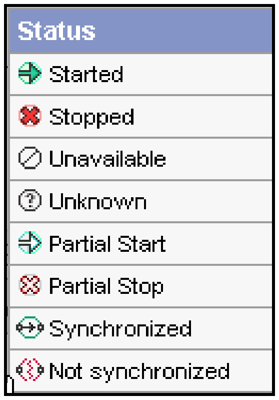


Figure 2-22 Status icons

If the server status is Unavailable, the node agent on the node where the application server is installed is not active. The server cannot be managed from the administrative console unless its node agent is active.

Using the command line

Distributed and z/OS platforms supply the serverStatus command in the *profile_root*/bin directory.

The syntax of the serverStatus command is:

```
serverStatus.bat(sh) <server>|-all [options]
```

Check a specific server with the serverStatus command as follows:

- For Windows: *profile_root*\bin\serverStatus *server_name*
- For UNIX and z/OS: *profile_root*/serverStatus.sh *server_name*

For i5/OS, issue:

```
WRKACTJOB SBS(QWAS61)
```

2.9.5 Start application servers

In a single server environment, you can use a command to start the application server.

In a managed environment, you can use a command in the system where the server runs or you can start the server from the administrative console of the deployment manager. However, the node agent for the application server node must be up before you can use the console.

Using the administrative console

To start one or more servers:

1. Select **Servers →Application Servers** to list the servers.
2. Select the server you want to start and click **Start**.

Application servers in a cluster can be managed as independent servers. A second option is to manage all the servers in the cluster using a single button. In the administrative console:

1. Select **Servers →Clusters**.
2. Select a cluster and select one of the following options:
   - **Start**: Use this option to start all servers in the cluster.
   - **Ripplestart**: Use this option to stop and start all servers in the cluster.

If you want to stop and restart all the application servers on a node::

1. From the administrative console, select **System Administration →Node Agents**.
2. Select the node agent.
3. Click **Restart all Servers on the Node**.

Using the command line

You can use the startServer command to start the application server on distributed and z/OS platforms:

- Windows: *profile_root*\bin\startServer *server_name*
- i5/OS: *profile_root*/bin/startServer *server_name*
- UNIX and z/OS: *profile_root*/bin/startServer.sh *server_name*

Starting a job (z/OS)

Use the following command to start the server:

```
START appserver_procname,JOBNAME=server_shortname,
ENV=cell_shortname.node_shortname.server_shortname
```

Part 3

Web services problem determination

This part contains the following IBM Redpaper:

- *WebSphere Application Server V6.1 Web Services Problem Determination*, REDP-4306 by Wendy Conti

3

Web services problem determination

Web services-related problems can occur when your application acts as a Web services client to a remote Web service or as a Web service that is accessed by external clients. Symptoms of a Web services problem would include:

- Web services errors and exceptions that appear in the JVM logs
- Unexpected Web services behavior
- Problems with Web services tooling, such as the Java2WSDL and WSDL2Java scripts
- Error or warning messages that begin with `WSWS`.

This paper covers problems that occur during development of Web services applications (WSDL2Java errors) and deployment of these applications to WebSphere Application Server. It also covers problems that occur in the Web services engine. It does not cover problems with Web services gateway, service integration bus, WSIF, or Web services security.

It is applicable to WebSphere Application Server on distributed platforms, i5/OS, and z/OS.

3.1 Introduction

WebSphere Application Server V6.1 provides extensive support for Web services standards. It can host both Web service client and provider applications. Problems can occur due to coding errors, invalid requests, deployment code generation, or the runtime configuration.

This activity covers the following types of Web services problems:

- Problems that occur during development
- Problems deploying Web services applications
- Problems that occur during runtime

If you believe that you have a Web services gateway, service integration bus, WSIF, or Web services security problem, go to “The next step” on page 155 for information about online resources and contacting IBM.

3.1.1 Determine problem type

Problems can occur during development, deployment, or runtime of Web services applications.

Symptoms of a problem during development

Two IBM products are targeted for development for WebSphere Application Server applications: Rational Application Developer and Application Server Toolkit.

Problems that occur during the development phase of Web service clients and providers for WebSphere Application Server are often a result of the use of the WSDL2Java tool. WSDL2Java is used for developing and deploying Java-based Web services applications from a WSDL file — both client and provider. This tool can be used stand-alone. It is also invoked by the development products and WebSphere deploy tools when working with Web services applications.

Symptoms of a WSDL2Java problem include error or warning messages with the WSWS prefix, or generated code that does not compile.

“Problems that occur during development” on page 113 focuses on problems that occur during the use of WSDL2Java when developing Web services applications.

Symptoms of a problem during deployment

Web services applications that are deployed to WebSphere Application Server require special deployment code to be generated. This code can be generated in a variety of ways, for example, with the WSDL2Java tool, by Rational Application

Developer, or by selecting an option during installation. However, the basis for each of these methods is, once again, the WSDL2Java tool.

Symptoms of a Web services deployment problem generally appear in the SystemOut log or, if using the **`wsdeploy`** command, in the command window. The first indication is a `WSWS0038E: Error from Web services deploy tool` message.

"Problems deploying Web services applications" on page 124 focuses on problems that occur when deploying a Web service client or provider application to WebSphere Application Server.

Symptoms of a problem during runtime

Problems that occur when running Web service clients or providers can happen for many reasons. Examples include (but are not limited to) an improper call to the provider, application code errors, runtime configuration errors, or a change in runtime conditions (for example, a provider URL that has changed).

Symptoms of a Web services runtime problem can appear on the client or the provider, either in a log or on the client display. How the problem presents will vary depending on factors such as where the problem occurred (client or provider), the type of client, how the application handles errors, and so on. Primary symptoms include messages with a WSWS prefix or unexpected results in the application.

"Problems that occur during runtime" on page 132 focuses on Web services problems that occur in WebSphere Application Server, where the application in WebSphere Application Server is the Web service provider or Web service client.

3.1.2 WSDL2Java

A WSDL file acts as a contract between client and provider. Client and provider agree to send SOAP messages structured according to what the WSDL describes. WSDL2Java is used for developing Java-based Web service client and provider applications from a WSDL file.

Web service provider applications can run in the EJB and Web container. Web service client applications can also run in the EJB and Web container, as well as the J2EE client container. These types of clients are commonly known as *managed clients*. A Web service client can also be a standalone Java application; this is commonly known as an *unmanaged client*.

But whether client or provider and whether it runs in a container or not, any application developed using WSDL2Java should comply with the JAX-RPC specification. It is when applications violate JAX-RPC that unexpected results occur.

WSDL2Java can be invoked directly by the developer. It runs behind the scenes in Rational Application Developer and the Application Server Toolkit when you develop Web services applications and prepare them for deployment. It is also used when you invoke WebSphere Application Server deploy tools for a Web services application.

WSDL2Java generates the following development code:

- The service endpoint interface
- Java types that map to JAX-RPC supported schema types defined/ referenced in the WSDL
- Exceptions that map to WSDL faults
- JAX-RPC mapping file that describes how the Java and XML are mapped to each other
- webservices.xml - the Web service provider deployment descriptor file
- ejb-jar.xml - the deployment descriptor for a Web service client running in the EJB container
- web.xml - the deployment descriptor for a Web service client running in the Web container
- WebSphere-proprietary deployment descriptor files
  - ibm-webservices-bnd.xmi and ibm-webservices-ext.xmi if provider
  - ibm-webservicesclient-bnd.xmi and ibm-webservicesclient-ext.xmi if client.

WSDL2Java also generates deployment code:

- The WebSphere-proprietary *_Helper, *_Deser, and *_Ser classes that bind the application to the engine and allow deserialization and serialization of messages. If the application contains exceptions, <Exception>_DeserProxy classes are also generated.
- If the application is a client, WebSphere-proprietary *ServiceInformation, *Locator, and *Stub classes.

Important: Do not modify deployment code that results from WSDL2Java. If you have modified the deployment code and are having problems, re-generate the deployment code and test the application without modification. Modified deployment code is not supported by IBM.

3.2 Problems that occur during development

This topic covers problems that occur as a result of using WSDL2Java during the development of Web service clients and providers for WebSphere Application Server.

WSDL2Java allows you to develop Java API for XML-based RPC (JAX-RPC) Web service applications. You do not have to use the WebSphere WSDL2Java tool to create a JAX-RPC application to run in WebSphere, but the application must comply with JAX-RPC Specification Version 1.1 and the Web Services for J2EE Specification Version 1.1.

- The JAX-RPC 1.1 specification can be downloaded at:

  `http://jcp.org/aboutJava/communityprocess/final/jsr101/index2.html`

- The Web services for J2EE specification (JSR 109) can be downloaded at:

  `http://java.sun.com/webservices/reference/apis-docs/index.jsp`

This activity does not include problems during runtime of applications developed with WSDL2Java. For that see “Problems that occur during runtime” on page 132.

3.2.1 Identify symptoms

Symptoms of a WSDL2Java problem covered in this topic include:

- Error or warning messages with the WSWS prefix occur when WSDL2Java is executed.

  When WSDL2Java is used from the command line, the error messages appear in the command window as it executes. When WSDL2Java errors result during wizard execution in Rational Application Developer, the error is displayed in a new error window.

- WSDL2Java generates code that does not compile.

  The compiler will report errors that occur during the compile of WSDL2Java-generated code. If you are using Rational Application Developer or the Application Server Toolkit, you will see the messages in the workbench. If using the command-line (javac), you will see the messages displayed to that interface (the DOS prompt or korn shell).

Symptoms discussed in the information center

You will find information about additional symptoms related to WSDL2Java in the WebSphere Application Server Information Center.

If you see your symptom in the following list, please refer to the relevant article in the Information Center.

- Compiler errors

  Web services compiled bindings troubleshooting tips:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rwbs_trbjavacompiler.html

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

- Command-line tools symptom list

  Web services command-line tools troubleshooting tips:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rwbs_trbcommand.html

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

    The following exception is displayed:

    ```
    System.InvalidOperationException: Method
    AnnuityInteropService.wsListAnnuityByHolder can not 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.
    ```

  – Multiprotocol port component restrictions with Web Services for J2EE Version 1.0 and 1.1 Specifications:

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

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

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

  – An emitter failure error occurs when running the WSDL2Java command on a WSDL document containing a JMS-style endpoint URL:

    ```
    WSWS3099E: Error: Emitter failure. Invalid endpoint address in
    port <x> in service <y>: <jms-url-string>
    ```

3.2.2 Collect and diagnostics

Collect the following diagnostic data:

- WSWSxxxxE error messages that resulted from issuing the WSDL2Java command
- Error messages issued as a result of using a Web services wizard in Rational Application Developer or Application Server Toolkit wizard (for example, the Web services client wizard)
- Output of the compilation (for example, messages that result from using javac)

3.2.3 Evaluate the error messages

Information about messages with the WSWS prefix, including suggested actions, can be found in the WebSphere Information Center in the Messages section under References. Review the user response for WSWS messages at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.webservices.multiprotocol.resources.mpMessages.html`

Note that this is the first of five WSWS message topics.

If you see a message other than WSWS3457W or WSWS3205E, go to "The next step" on page 155 for information about performing online searches and gathering information required by IBM support.

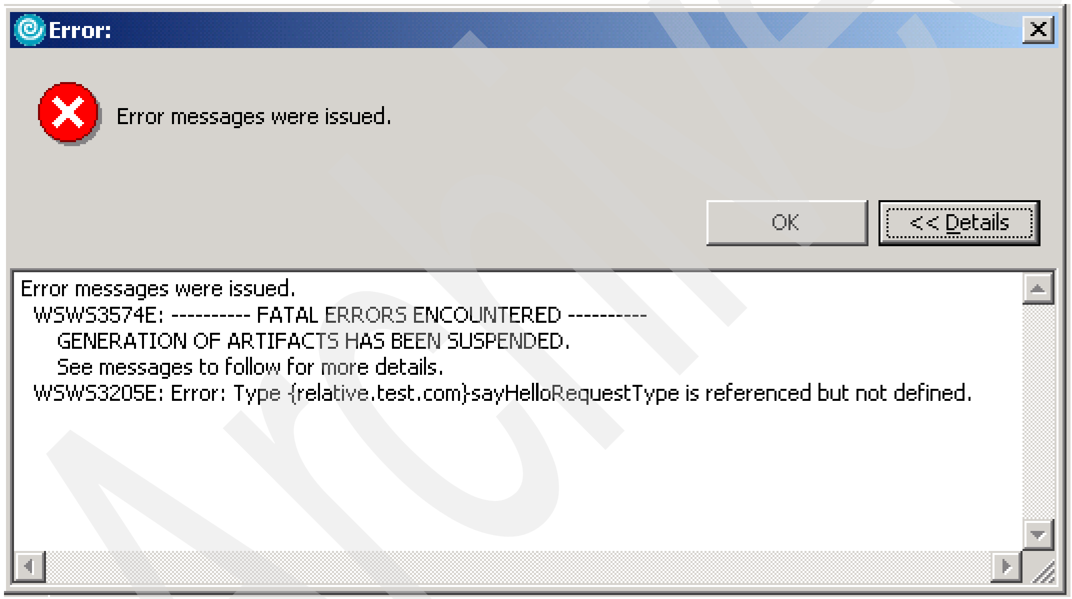
WSWS3574E, WSWS3205E

The messages shown in Example 3-1 indicate that a relative URI has been used. If you see these messages, go to "targetNamespace is a relative URI" on page 120.

Example 3-1   WSWS3574E, WSWS3205E

```
WSWS3574E: ---------- FATAL ERRORS ENCOUNTERED ----------
   GENERATION OF ARTIFACTS HAS BEEN SUSPENDED.
   See messages to follow for more details.
WSWS3205E: Error: Type {relative.test.com}sayHelloRequestType is
referenced but not defined.
```

The same error in Rational Application Developer would appear during the execution of a Web services wizard. It would appear in an error box (Figure 3-1).



```
Error messages were issued.
 WSWS3574E: ---------- FATAL ERRORS ENCOUNTERED ----------
   GENERATION OF ARTIFACTS HAS BEEN SUSPENDED.
   See messages to follow for more details.
 WSWS3205E: Error: Type {relative.test.com}sayHelloRequestType is referenced but not defined.
```

Figure 3-1   WSWS3547E issued in Rational Application Developer

WSWS3099E

If you have the emitter failure error `WSWS3099E: Error: Emitter failure. Invalid endpoint address in port <x> in service <y>: <jms-url-string>` when running the WSDL2Java command on a WSDL document containing a JMS-style endpoint URL, see the “Web services command-line tools troubleshooting tips” WebSphere Information Center article for more information, available at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rwbs_trbcommand.html`

3.2.4 Evaluate compiler output

Compiler output should be scanned for indications of an error. Some errors are self-explanatory, but others may require a search for known problems.

Illegal start of type

Example 3-2 shows an example of compiler output with errors. In this example, the compiler is complaining about the period character immediately preceding TestHelloResponse, TestHelloRequest, and Test.

This particular problem was caused because a non-traditional URI was used in the WSDL targetNamespace. For more information about the use of non-traditional URIs, see “targetNamespace contains a non-traditional URI scheme” on page 118.

Example 3-2 Compiler output with errors

```
Test.java:10: illegal start of type
    public .TestHelloResponse testHello(.TestHelloRequest parameters)
throws java.rmi.RemoteException, .HelloWorldException;
           ^
Test.java:10: <identifier> expected
    public .TestHelloResponse testHello(.TestHelloRequest parameters)
throws java.rmi.RemoteException, .HelloWorldException;

^
TestService.java:10: illegal start of type
    public .Test getTest() throws javax.xml.rpc.ServiceException;
           ^
TestService.java:10: <identifier> expected
    public .Test getTest() throws javax.xml.rpc.ServiceException;
                                                                ^
TestService.java:14: illegal start of type
```

```
    public .Test getTest(java.net.URL portAddress) throws
javax.xml.rpc.ServiceException;
           ^
TestService.java:14: <identifier> expected
    public .Test getTest(java.net.URL portAddress) throws
javax.xml.rpc.ServiceException;

^
6 errors
```

Where to go from here: If you have not identified a specific symptom in those discussed here, go to “The next step” on page 155 for information about performing an online search of Web services problems and for a list of useful technical articles. If these resources do not help you resolve the problem, contact IBM support.

3.2.5 targetNamespace contains a non-traditional URI scheme

If a WSDL schema targetNamespace contains a non-traditional URI scheme, WSDL2Java can generate code that does not compile.

For information about URI syntax see *RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax* at:

http://www.faqs.org/rfcs/rfc2396.html

Example 3-3 shows an example of WSDL that contains a non-traditional URI in its namespace declarations and embedded schema targetNamespace.

Example 3-3 WSDL containing non-traditional URI

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="hhttp://test.com"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:intf="hhttp://test.com"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
   <schema targetNamespace="hhttp://test.com"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <element name="testHelloResponse">
         <complexType>
           <sequence>
             <element name="testHelloReturn" nillable="true" type="xsd:string"/>
           </sequence>
         </complexType>
```

```
        </element>
        <element name="testHelloRequest">
          <complexType>
            <sequence>
             <element name="name" nillable="true" type="xsd:string"/>
            </sequence>
          </complexType>
      </element>
</schema>
```

Resolve the problem

Previously, WSDL2Java did not consider non-traditional URI schemes when mapping namespaces to package names. PK35427 fixes WSDL2Java so that it can properly parse namespaces with such non-traditional URI schemes and transform them into package names, instead of just an empty string.

The solution is to apply fix pack 6.1.0.9, which includes PK35427, on the WebSphere installation where WSDL2Java is invoked. Then invoke WSDL2Java to re-generate the classes. The newly generated classes should not contain any compilation errors.

Workaround

To work around the compilation error, change the WSDL and schema targetNamespace to use a traditional URI scheme, such as http, file, or urn. For example, from:

```
hhttp://test.com
```

to:

```
http://test.com
```

Then invoke WSDL2Java to re-generate the classes. The resulting classes should not contain any compilation errors.

More about this problem

Consider the WSDL schema shown in Example 3-3 on page 118. Example 3-4 shows the portion of the WSDL that references these schema types.

Example 3-4 WSDL referencing the schema

```
<wsdl:message name="testHelloRequest">
    <wsdl:part name="parameters" element="intf:testHelloRequest"/>
  </wsdl:message>
  <wsdl:message name="testHelloResponse">
```

```
    <wsdl:part name="parameters" element="intf:testHelloResponse"/>
  </wsdl:message>
  <wsdl:portType name="Test">
    <wsdl:operation name="testHello">
      <wsdl:input name="testHelloRequest" message="intf:testHelloRequest"/>
      <wsdl:output name="testHelloResponse" message="intf:testHelloResponse"/>
    </wsdl:operation>
  </wsdl:portType>
```

From this WSDL, WSDL2Java would map the complexTypes of elements testHelloRequest and testHelloResponse to Java types, and generate them in no package. Hence, any other classes that reference these Java types would incorrectly use the notation shown in Example 3-5.

Example 3-5 WSDL2Java generated code that will not compile

```
public interface Test extends java.rmi.Remote {
    public .TestHelloResponse testHello(.TestHelloRequest parameters)
    throws java.rmi.RemoteException;
}
```

The Java compiler would complain about the period character immediately preceding the Java types TestHelloResponse and TestHelloRequest.

Support preparation

If the work around or applying fix pack 6.1.0.9 does not resolve the problem, collect the following data for use by IBM support:

- WSDL and any referenced schema files
- WSDL2Java command invocation, or a description of how Rational Application Developer or the Application Server Toolkit was used to generate the non-compiling classes

Go to “The next step” on page 155 for information about performing an online search of Web services problems and for a list of useful technical articles. If these resources do not help you resolve the problem, contact IBM support.

3.2.6 targetNamespace is a relative URI

The error is not due to a product defect. It occurs when the WSDL contains an embedded schema with a targetNamespace set to a relative URI. The FATAL ERROR message just indicates that WSDL2Java will stop generating code. It does not indicate a fatal error with the operating system — just that WSDL2Java will abort.

Resolve the problem

Note: WSDL2Java requires that targetNamespaces be absolute URIs.

Change the WSDL so that it uses absolute URIs and repeat the WSDL2Java process.

More about this problem

When invoking WSDL2Java on the command line, you will get an error similar to Example 3-6 displayed to the console.

Example 3-6 WSWS3574E, WSWS3205E messages

```
WSWS3574E: ---------- FATAL ERRORS ENCOUNTERED ----------
   GENERATION OF ARTIFACTS HAS BEEN SUSPENDED.
   See messages to follow for more details.
WSWS3205E: Error: Type {relative.test.com}sayHelloRequestType is
referenced but not defined.
java.io.IOException: FATAL ERRORS encountered by WSDL2Java tool.
   at com.ibm.ws.webservices.wsdl.Parser.generate(Parser.java:467)
   at com.ibm.ws.webservices.wsdl.Parser.generate(Parser.java:394)
   at com.ibm.ws.webservices.wsdl.Parser.run(Parser.java:299)
   at com.ibm.ws.webservices.wsdl.WSDL2.run(WSDL2.java:318)
   at com.ibm.ws.webservices.tools.WSDL2Java.main(WSDL2Java.java:492)
   at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
   at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.ja
va:64)
   at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccesso
rImpl.java:43)
   at java.lang.reflect.Method.invoke(Method.java:615)
   at com.ibm.ws.bootstrap.WSLauncher.main(WSLauncher.java:263)
```

This error happens because the WSDL contains an embedded schema with a targetNamespace set to a relative URI.

Example

In this example, the WSDL definitions element contains the namespace declarations shown in Example 3-7.

Example 3-7 WSDL namespace declarations

```
<wsdl:definitions targetNamespace="http://helloworld.com"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns1="relative.test.com"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 ...>
```

Then the WSDL embeds an XML schema (Example 3-8).

Example 3-8 Embedded XML schema

```
<wsdl:types>
 <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="relative.test.com">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="sayHelloRequestType">
      <sequence>
         <element name="msg" type="xsd:string"/>
         <element name="id" type="xsd:int"/>
      </sequence>
    </complexType>
    <complexType name="sayHelloResponseType">
      <sequence>
        <element name="msg" type="xsd:string"/>
    <element name="id" type="xsd:int"/>
      </sequence>
    </complexType>
  </schema>
</wsdl:types>
```

Notice the embedded schema's targetNamespace, relative.test.com. The schema complexTypes are then referenced later in the WSDL messages, as shown in Example 3-9.

Example 3-9   WSDL messages

```
<wsdl:message name="sayHelloRequest">
        <part name="parameter" type="tns1:sayHelloRequestType"/>
  </wsdl:message>
  <wsdl:message name="sayHelloResponse">
    <wsdl:part name="parameter" type="tns1:sayHelloResponseType"/>
  </wsdl:message>
```

WSDL2Java requires that targetNamespaces be absolute URIs. Since the WSDL has a targetNamespace of http://helloworld.com and the embedded schema has a specified targetNamespace of relative.test.com, WSDL2Java tries to make that relative URI absolute by appending it to the base URN — the WSDL targetNamespace.

In other words, given this WSDL targetNamespace http://helloworld.com and schema targetNamespace relative.test.com, WSDL2Java would resolve the targetNamespace of the embedded schema as:

```
http://helloworld.comrelative.test.com
```

However, the prefix tns1 is set to the relative URI relative.test.com, and the schema complexTypes are referred to using that prefix shown (see Example 3-9).

In essence, tns1:sayHelloRequestType means the same as {relative.test.com}sayHelloRequestType.

The WSDL messages reference the schema types using namespace relative.test.com, but WSDL2Java resolved the targetNamespace of those schema types as http://helloworld.comrelative.test.com. Therefore, it issues the error message:

```
WSWS3205E: Error: Type {relative.test.com}sayHelloRequestType is
referenced but not defined.
```

One way to solve the error is to set the tns1 prefix to the expected URI, http://helloworld.comrelative.test.com, as shown in Example 3-10.

Example 3-10 Modified namespace declaration

```
<wsdl:definitions targetNamespace="http://helloworld.com"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns1="http://helloworld.comrelative.test.com"

 ...>
```

But, the namespace http://helloworld.comrelative.test.com is most likely not what was intended. So the best guideline is to avoid relative URIs altogether, as in Example 3-11.

Example 3-11 targetNamespace modified to be absolute URI

```
   <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://relative.test.com">
       …
     </schema>
   </wsdl:types>
```

Support preparation

If changing the schema targetNamespace URI from relative to absolute does not alleviate the WSWS3205E error, collect the following data for use by IBM support:

- WSDL
- WSDL2Java command invocation, or a description of how Rational Application Developer or the Application Server Toolkit was used when the error was encountered

Go to “The next step” on page 155 for information about performing an online search of Web services problems and for a list of useful technical articles. If these resources do not help you resolve the problem, contact IBM support.

3.3 Problems deploying Web services applications

This activity covers Web services problems that occur when deploying a Web service client or provider application to WebSphere Application Server. Specifically, it covers problems that occur in WSDL2Java.

Important: Do not modify deployment code that results from WSDL2Java. If you have modified the deployment code and are having problems, re-generate the deployment code and test the application without modification. Modified deployment code is not supported by IBM.

A Web services application deployed to WebSphere Application Server must include the various, proprietary deployment code generated by WSDL2Java. This deployment code can be generated by:

- The WSDL2Java tool
- Selecting the -deployws option when installing the application using wsadmin
- Selecting the Deploy WebServices option when installing the application using the administrative console
- The wsdeploy tool
- Using Rational Application Developer or the Application Server Toolkit

However, keep in mind that all of these methods use WSDL2Java to generate the code.

The WSDL2Java options to deploy the application depend on the application type:

- If the application is a Web service implementation, then WSDL2Java is invoked with the -role deploy-server option.
- If the application is a Web service client, then WSDL2Java is invoked with the -role deploy-client option.
- For both client and server role, the -introspect option is also enabled.

3.3.1 Identify symptoms

Symptoms of a Web services deployment problem generally appear in the SystemOut log or, if using the **`wsdeploy`** command, in the command window. The first indication is the following message:

```
WSWS0038E: Error from Web services deploy tool
```

This message is followed by more specific error messages in the following format:

```
WSWSxxxxE
```

Where to go: If you are experiencing WSWS3467E, go to “targetNamespace contains a non-traditional URI scheme” on page 118.

If you do not know the message or are experiencing any other symptoms related to Web services deployment, go to “Collect diagnostics” on page 126.

3.3.2 Collect diagnostics

Collect SystemOut log. For information about collecting SystemOut see “Collecting JVM logs” on page 152.

3.3.3 Analyze SystemOut

Scan the SystemOut log for WSWSxxxxE messages. See Example 3-12.

Example 3-12 WSWSxxxxE messages in SystemOut

```
[1/19/07 8:59:24:984 MST] 00000065 WSDeployTask  E   WSWS0038E: Error
from Web services deploy tool:
WSWS3574E: ---------- FATAL ERRORS ENCOUNTERED ----------
   GENERATION OF ARTIFACTS HAS BEEN SUSPENDED.
   See messages to follow for more details.

.. error messages

[1/19/07 8:59:25:000 MST] 00000065 WSDeployTask E WSWS0038E: Error from
Web services deploy tool: Error: FATAL ERRORS encountered by WSDL2Java
tool.
```

Evaluate SystemOut

Information about messages with the WSWS prefix, including suggested actions, can be found in the WebSphere Information Center in the Messages section under References.

If you see `WSWS3467E: WSDL2Java validation error`, go to “targetNamespace contains a non-traditional URI scheme” on page 118.

Otherwise, go to “The next step” on page 155 for information about performing online searches and gathering information required by IBM support.

3.3.4 Invalid public constructor

WSWS3467E occurs when the validation process that occurs during the WSDL2Java command has detected an invalid constructor. Specifically, it has detected an exception that is missing a constructor that takes the same number of parameters that correspond to the number of elements inside of the element's complexType.

The WSDL2Java command (invoked during the deploy process) uses the -introspect option, which directs the WSDL2Java command to examine and validate existing Java classes against the JAX-RPC 1.1 specification and the WSDL, and to generate code that is compatible with those existing Java classes.

WSWS3467E errors are a result of an error detected by this validation.

Resolve the problem

Correct the constructor and re-deploy the application.

More about this problem

The JAX-RPC 1.1 specification describes Java APIs for XML-based Remote Procedure Call (RPC) mechanisms. It is one of the central specifications that the Web services engine follows. Its description of WSDL and XML to Java mapping is crucial to understanding and resolving this problem.

Example

In this example, the user receives the error message shown in Example 3-13.

Example 3-13 WSWS3467E

```
[1/19/07 8:59:24:984 MST] 00000065 WSDeployTask  E   WSWS0038E: Error
from Web services deploy tool:
WSWS3574E: ---------- FATAL ERRORS ENCOUNTERED ----------
   GENERATION OF ARTIFACTS HAS BEEN SUSPENDED.
   See messages to follow for more details.

[1/19/07 8:59:24:984 MST] 00000065 WSDeployTask E WSWS0038E: Error from
Web services deploy tool: WSWS3467E: ---------- WSDL2Java VALIDATION
ERROR ----------
        Existing Fault class is invalid: "com.test.HelloWorldException"
        Does not implement a valid public constructor:
"HelloWorldException", containing 3 parameters.

[1/19/07 8:59:25:000 MST] 00000065 WSDeployTask E WSWS0038E: Error from
Web services deploy tool: Error: FATAL ERRORS encountered by WSDL2Java
tool.
```

First, examine HelloWorldException, the exception that the message claims is invalid. See Example 3-14.

Example 3-14 HelloWorldException

```
package com.test;

public class HelloWorldException extends java.lang.Exception {
  private java.lang.String message;
  private java.lang.Integer id;
  private java.lang.String detail;

  public HelloWorldException(java.lang.String message) {
    this.message = message;
  }

  public java.lang.String getMessage() {
    return message;
  }

  public java.lang.Integer getId() {
    return id;
  }
  public java.lang.String getDetail() {
    return detail;
  }

}
```

The error message specifically mentions the exception's constructor. Notice that the constructor takes one parameter. Yet the message states that in order for the constructor to be valid, the constructor must take three parameters. What is the basis for this claim?

Examine the WSDL file. Remember, when a Web services application is deployed, WSDL2Java is used *behind the scenes* to examine existing classes in the application, validate them against the WSDL, and verify that they comply with JAX-RPC.

That schema type is an anonymous complexType and is defined as shown in Example 3-15.

Example 3-15 hellowWorldException element - anonymous complexType definition

```
<element name="helloWorldException">
    <complexType>
     <sequence>
   <element name="message" nillable="true" type="xsd:string"/>
   <element name="id" nillable="true" type="xsd:int"/>
   <element name="detail" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
```

The helloWorldException element is referenced in the message with the same name, as shown in Example 3-16.

Example 3-16 helloWorldException message defined in WSDL

```
<wsdl:message name="helloWorldException">
    <wsdl:part name="fault" element="impl:helloWorldException"/>
  </wsdl:message>
```

And this message is referenced as a fault in the WSDL operation, testHello, as shown in Example 3-17.

Example 3-17 WSDL operation containing WSDL fault

```
<wsdl:operation name="testHello">
  <wsdl:input name="testHelloRequest"
         message="impl:testHelloRequest"/>
  <wsdl:output name="testHelloResponse"
         message="impl:testHelloResponse"/>
  <wsdl:fault name="helloWorldException"
         message="impl:helloWorldException"/>
</wsdl:operation>
```

Note: The JAX-RPC specification says this about the mapping of WSDL faults to Java exceptions:

> A wsdl:fault is mapped to either a java.rmi.RemoteException (or its subclass), service specific Java exception (described later in this section) or a javax.xml.rpc.soap.SOAPFaultException.

JAX-RPC 1.1 goes on to elaborate on service specific exceptions:

> A service specific Java exception (mapped from a wsdl:fault and the corresponding wsdl:message) extends the class java.lang.Exception directly or indirectly. The single message part in the wsdl:message (referenced from the wsdl:fault element) may be either a type or an element. If the former, it can be either a xsd:complexType or a simple XML type. **Each element inside the xsd:complexType is mapped to a getter method and a parameter in the constructor of the Java exception. Mapping of these elements follows the standard XML to Java type mapping**.

The full specification can be downloaded from:

`http://java.sun.com/xml/downloads/jaxrpc.html`

In the case of our example, the WSDL fault is clearly mapped to a service-specific exception. The bolded text highlights the remarks relevant to the WSWS3467E WSDL2Java validation error. The HelloWorldException class is missing a constructor that takes three parameters that correspond to the three elements inside of the helloWorldException element's complexType.

This point is illustrated by contrasting the exception constructor (Example 3-18) with its mapped complexType (Example 3-19).

Example 3-18   HelloWorldException class constructor - incorrect

```
public HelloWorldException(java.lang.String message) {
    this.message = message;
    }
```

Example 3-19   helloWorldException element and complexType definition

```
<element name="helloWorldException">
    <complexType>
     <sequence>
   <element name="message" nillable="true" type="xsd:string"/>
   <element name="id" nillable="true" type="xsd:int"/>
   <element name="detail" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
```

To be compliant with JAX-RPC, HelloWorldException should also implement a constructor like that shown in Example 3-20.

Example 3-20   HelloWorldException class constructor - correct

```
public HelloWorldException(
           java.lang.String message,
           java.lang.Integer id,
           java.lang.String detail) {
        this.message = message;
        this.id = id;
        this.detail = detail;
    }
```

Support preparation

If changing the exception constructor does not resolve the problem, collect the following data for use by IBM support:

- The complete Web services application that fails to deploy, including the WSDL file and the Java source of the service specific exception.
- A description of how the Web services application was deployed — for instance, invoking the wsdeploy tool or installing the application using the administrative console.

- If deploying the application using the administrative console, enable the Web services trace to capture the WSWS3467E deployment error. Instructions for enabling the Web services trace can be found in “Enabling the Web services trace” on page 154.
- If wsdeploy is used, then collect a simple text capture of the error sent to standard output.

Go to “The next step” on page 155 for information about performing an online search of Web services problems and for a list of useful technical articles. If these resources do not help you resolve the problem, contact IBM support.

3.4 Problems that occur during runtime

This topic covers problems that occur in Web services applications running in WebSphere Application Server.

- WebSphere application as the Web service client

  When a WebSphere application invokes a Web service, it becomes a Web service client. The Web service being invoked can be another WebSphere application or may be hosted on any other type of system that provides Web services. The system hosting the Web service is the Web service provider.

  Problems invoking a Web service can be a result of several things, for example:

  – The application acting as the Web service provider is not available or is broken.

  – The URL for the Web service provider has changed.

  – The Web service client code has a problem.

  – The Web service request is routed through a gateway or enterprise service bus with a problem.

- WebSphere application as the Web service provider

  When a WebSphere application that provides a Web service experiences a problem, it may behave incorrectly or become unavailable. In addition to any error symptoms generated by the application, this type of problem usually creates error symptoms at the Web service client. The Web service client may or may not be another WebSphere application.

3.4.1 Check system integrity

When a WebSphere Application Server application is the Web service client and receives an error, the simplest thing to do initially is to ensure that the Web service provider is available and that the URL used by the requester is correct.

Clients attempting to access a provider with an incorrectly specified URL commonly occurs when applications are moved from development environments to production ones. Ports and host names most likely change, and clients must be updated with the new Web service target URL.

Tip: When the Web service provider is an application running in WebSphere Application Server, you can test the availability of the service by pointing a browser to the URL. If the test is successful, you should see two lines. The first contains the full QName of WSDL port, and the second the message `Hi there, this is a Web service!`

3.4.2 Identify symptoms

Symptoms of a Web services problem appear in a variety of ways depending on the problem, the client's environment, and how the client was designed. For example, if invoking an unmanaged client from the command-line console, the error messages would be displayed to that interface. If the client is a Web application that writes to an output stream that appears in the Web browser, then the errors would be displayed to the browser. On the provider system, the messages will appear in the application server SystemOut log.

Messages will, with a WSWS prefix indicate a Web services related error. Problems may also occur that do not produce a Web services error message but will appear as unexpected results in the application. A trace of the SOAP messages often shows the error.

Specific symptoms covered in this activity include:

- Client system
  - javax.xml.rpc.ServiceException with message code WSWS5055E or WSWS5014E.

    For more information about this symptom see "targetNamespace contains a non-traditional URI scheme" on page 118.
  - WSWS3713E: Connection to the remote host localhost failed. Received the following error: `Connection refused`.

    Verify that the host and port used to call the provider are correct.

- Client or provider system
  - org.xml.sax.SAXException: WSWS3047E

    For more information about this symptom see "Improperly formed SOAP request or response" on page 146.
  - A SOAP message with an unexpectedly empty body

    For more information about this symptom see "Serialized SOAP message does not contain children in body" on page 143.

3.4.3 Collect diagnostics

If the symptoms are not apparent, you will need to collect the following:

- If the problem occurs at the client:
  - Messages displayed.
  - Logs that the client application writes messages to.
  - If the client is a managed client running in the server, collect SystemOut log.
- If the problem occurs at the provider: SystemOut log.

  For information about collecting the SystemOut log see "Collecting JVM logs" on page 152.
- TCPMonitor trace.

  Using TCPMonitor to view messages will not always be applicable or relevant in every case. Sometimes, runtime problems occur before the SOAP message can be serialized and sent across the wire. Or problems may have nothing to do with how a message appears.

  However, for certain problems, it is important to have a copy of the SOAP request/response exchange. In particular, capture the trace when:
  - There is an issue with an HTTP header, for example, it is not set to the expected value or it does not appear at all.
  - There appears to be a deserialization problem. The message text or Java stack trace usually contains the word *deserialization* if this is the case.
  - There is an issue with a SOAP message.

  The message exchange can be captured by recreating the problem and tracing it using TCPMonitor. For a quick reference on using TCPMonitor, see "Collecting trace data with TCPMonitor" on page 153.

  You may want to analyze the SystemOut log first before deciding whether you should recreate the problem and collect this trace.

Glossary terms:

- *Serialization* is the process of converting Java objects to XML.
- *Deserialization* is the process of converting XML to Java objects.

3.4.4 Analyze diagnostics

To analyze the diagnostics:

1. Scan the SystemOut log for errors.
2. If captured, scan the TCPMonitor trace output for the message causing the problem.

If your specific symptom is not addressed here, go to "The next step" on page 155 for information about performing online searches and gathering information required by IBM support.

Analyze SystemOut

In the SystemOut log, look for and make a note of any of the following:

- Error messages with WSWS as the prefix.

  Review the user response for WSWS messages at:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.webservices.multiprotocol.resources.mpMessages.html`

  Note that this is the first of five WSWS message topics.

- javax.xml.rpc.ServiceException
- org.xml.sax.SAXException
- Java stack trace with any other errors

javax.xml.rpc.ServiceException, WSWS5055E, WSWS5014E

The following messages indicate a problem when using the *wsdl_service*Locator class generated by WSDL2Java to obtain a stub object for invoking the Web service.

- `javax.xml.rpc.ServiceException: WSWS5055E: Cannot find a Service for namespace null`
- `javax.xml.rpc.ServiceException: WSWS5014E: An error occurred instantiating the generated Stub class class: java.lang.reflect.InvocationTargetException`

Prior to fix pack 6.0.2.19 for V6.0.2 and fix pack 6.1.0.9 for V6.1, this can be due to a bug in the WSDL2Java tool that occurs when port names contain non-alphanumeric characters.

For more information see “Port name problem” on page 139.

org.xml.sax.SAXException: WSWS3047E

WSWS3047E can appear on the client or provider system. If you see this message see “Improperly formed SOAP request or response” on page 146.

When a Web service client sends a request to a WebSphere Application Server Web service, you might observe a SOAP fault like in Example 3-21 as a response.

Example 3-21 WSWS3047E SOAP fault

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.generalException</faultcode>
         <faultstring><![CDATA[org.xml.sax.SAXException:
            WSWS3047E: Error: Cannot deserialize element name of bean
             com.test.TestHello.
            Message being parsed:
            <soapenv:Envelope
                    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
               <soapenv:Header/>
                 <soapenv:Body>
                  <p585:testHello xmlns:p585="http://test.com">
                    <p585:name>Wendy</p585:name>
                  </p585:testHello>
                 </soapenv:Body>
             </soapenv:Envelope>]]>
         </faultstring>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

On the WebSphere Web service provider, the error might be manifested in the server's SystemOut. See Example 3-22.

Example 3-22 WSWS3047E on the Web service provider

```
[3/7/07 16:43:13:708 CST] 00000022 UserException E   WSWS3228E: Error: Exception:
WebServicesFault
 faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.generalException
 faultString: org.xml.sax.SAXException: WSWS3047E: Error: Cannot deserialize element
name of bean com.test.TestHello.

Message being parsed:
<soapenv:Envelope  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <p585:testHello xmlns:p585="http://test.com">
      <p585:name>Wendy</p585:name>
    </p585:testHello>
  </soapenv:Body>
</soapenv:Envelope>

faultActor: null faultDetail:

org.xml.sax.SAXException: WSWS3047E: Error: Cannot deserialize element name of bean
com.test.TestHello. Message being parsed:
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
    <p585:testHello xmlns:p585="http://test.com">
      <p585:name>Wendy</p585:name>
    </p585:testHello>
  </soapenv:Body>
</soapenv:Envelope>
```

Similarly, the WSWS3047E error might occur on the client, if it is running in the WebSphere environment.

If the WebSphere Web service client is a managed one running in the Web or EJB container, you might also see in the server's SystemOut a stack trace beginning with text similar to Example 3-23.

Example 3-23 WSWS3047E on the Web service client

```
? 00000028 MCUtils       1 com.ibm.ws.webservices.engine.utils.MCUtils
interceptDeserializationException interceptDeserializationException:
org.xml.sax.SAXException: WSWS3047E: Error: Cannot deserialize element
name of bean com.test.TestHello. org.xml.sax.SAXException: WSWS3047E:
Error: Cannot deserialize element name of bean com.test.TestHello.
```

If the client is an unmanaged one and using the WebSphere environment, you might see the stack trace displayed in standard out or standard err (such as the command-line console). During development cycles, developers commonly use Rational Application Developer to invoke unmanaged test clients, and might see the WSWS3047E stack trace displayed in the RAD console.

Where to go from here:

- If the problem appears to involve deserialization, the SOAP message format, or the HTTP header, collect a TCPMonitor trace of the problem.
- If you found errors not listed here, go to “The next step” on page 155.

Analyze TCPMonitor output

To analyze the TCPMonitor output:

1. Scan the output to find the message that caused the problem.
2. Examine the message to make sure that it is formed properly according to the WSDL.

Empty SOAP body

A SOAP request or response with unexpected content can cause problems in a Web service client or provider.

One problem that you may experienced is a SOAP message with an unexpectedly empty body, described by a WSDL that defines a document/literal message with no parts. If you have this situation, go to “Serialized SOAP message does not contain children in body” on page 143. See Example 3-24.

Example 3-24 Message with empty SOAP Body

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body/>
</soapenv:Envelope>
```

3.4.5 Port name problem

WSWS5055E and WSWS5014E can indicate a reported error in the WSDL2Java tool that occurs when the WSDL port name contains characters other than alphanumerics (for example, periods or dashes).

Resolve the problem

This problem was reported by APAR PK35399 and is fixed in fix pack 6.1.0.9 for V6.1 and fix pack 6.0.2.19 for V6.0.2. Under this APAR, WSDL2Java was changed so that it generated a correct *wsdl_service*Locator class.

Apply the fix pack, then re-generate client deployment code, specifically, the *wsdl_service*Locator class.

Quick test

As a test to determine whether your problem is the same as the problem discussed here, you can modify the *wsdl_service*Locator class, bearing in mind that doing so is not an approved practice and just serves as a temporary work around until the true solution can be applied.

The *wsdl_service*Locator.getPort2NamespaceMap() method should be modified so that the XML QName local part of the WSDL port name is used as the key to the namespace value.

For example, given the following WSDL port definition:

```
<wsdl:port name="Test-Service" binding="impl:TestSoapBinding">
<wsdlsoap:address
location="http://localhost:9080/TestHello/Test"/>
</wsdl:port>
```

change from:

```
port2NamespaceMap.put(
"TestService",
"http://schemas.xmlsoap.org/wsdl/soap/");
```

to:

```
port2NamespaceMap.put(
"Test-Service",
"http://schemas.xmlsoap.org/wsdl/soap/");
```

Save this modified class and then re-invoke client. The WSWS5055E error should no longer occur.

> **Important:** This is a temporary check only. If the client is re-deployed, then this modified class will be overwritten.

Support preparation

If applying the fix pack and re-generating client deployment code does not resolve the problem, collect the following data for use by IBM support:

- Web services trace containing the javax.xml.rpc.ServiceException: WSWS5055E or WSWS5014E error. See "Enabling the Web services trace" on page 154 for instructions on collecting the trace.
- Client application test case, including the WSDL file and Java source.

Go to "The next step" on page 155 for information about performing an online search of Web services problems and for a list of useful technical articles. If these resources do not help you resolve the problem, contact IBM support.

More about this problem

Consider the WSDL file excerpt in Example 3-25.

Example 3-25 WSDL service definition

```
<wsdl:service name="Test-Service">
    <wsdl:port name="Test-Service" binding="impl:TestSoapBinding">
      <wsdlsoap:address
location="http://localhost:9080/TestHello/Test"/>
    </wsdl:port>
  </wsdl:service>
```

When invoked to generate client development code, WSDL2Java generates an interface that extends the javax.xml.rpc.Service interface, and that maps to the WSDL service. In this example, that interface is called com.test.TestService, named after the WSDL service.

When emitting client deployment code, WSDL2Java emits a WebSphere proprietary class that implements this interface. The convention used in naming this class is the name of the interface mapped to WSDL service (again, in this example, com.test.TestService) followed by *Locator* — in this example, com.test.TestServiceLocator.

You can use the *wsdl_service*Locator class to obtain a client-side stub implementation of the remote service endpoint.

Example 3-26 shows how a client application might use the *wsdl_service*Locator class to obtain the stub.

Example 3-26 Using the *wsdl_service*Locator class to obtain the stub

```
import com.test.Test;
import com.test.TestServiceLocator;
try {
Test stub = new TestServiceLocator().getTestService();
...
```

Test is the service endpoint interface (what will actually be returned is an instance of another WSDL2Java-generated deployment class, TestSoapBindingStub, which implements Test).

Examine the javax.xml.rpc.ServiceException stack trace in Example 3-27.

Example 3-27 javax.xml.rpc.ServiceException stack trace

```
Caught exception javax.xml.rpc.ServiceException: WSWS5055E: Cannot find
a Service for namespace null.
javax.xml.rpc.ServiceException: WSWS5055E: Cannot find a Service for
namespace null.
   at
com.ibm.ws.webservices.multiprotocol.utils.ServiceManager.getServiceFor
Namespace(ServiceManager.java:105)
   at
com.ibm.ws.webservices.multiprotocol.AgnosticService.getGeneratedStub(A
gnosticService.java:525)
   at
com.ibm.ws.webservices.multiprotocol.AgnosticService.doGetPort(Agnostic
Service.java:462)
   at
com.ibm.ws.webservices.multiprotocol.AgnosticService.getStub(AgnosticSe
rvice.java:405)
   at
com.test.TestServiceLocator.getTestService(TestServiceLocator.java:68)
   at
com.test.TestServiceLocator.getTestService(TestServiceLocator.java:63)
   at com.test.TestClient.main(TestClient.java:12)
```

Since it is referenced in a stack frame in the javax.xml.rpc.ServiceException stack trace, line 68 of TestServiceLocator might offer possible clues. Line 68, shown in Example 3-28, is a call from getTestService() to getStub().

Example 3-28 Call from getTestService() to getStub()

```
com.test.Test _stub =
            (com.test.Test) getStub(
                testServicePortName,
                (String)
                getPort2NamespaceMap().get(testServicePortName),
                com.test.Test.class,
                "com.test.TestSoapBindingStub",
                portAddress.toString());
```

getStub() is implemented in a parent class of TestServiceLocator, com.ibm.ws.webservices.multiprotocol.AgnosticService. There is never any need for you to examine internal engine code. However, for the purposes of understanding this problem, it is meaningful to know that the second parameter that AgnosticService.getStub() takes is the namespace of the WSDL port.

The way that getTestService() obtains that namespace is by calling the getPort2NamespaceMap() method, implemented in TestServiceLocator:

```
...(String) getPort2NamespaceMap().get(testServicePortName)...
```

The testServicePortName variable is declared and initialized earlier in the class:

```
private java.lang.String testServicePortName = "Test-Service";
```

Notice how the string Test-Service matches the value of the WSDL port, defined as a child of the WSDL service:

```
<wsdl:port name="Test-Service" … >
```

Now look at the implementation of the getPort2NamespaceMap() method in Example 3-29.

Example 3-29 getPort2NamespaceMap()

```
private java.util.Map port2NamespaceMap = null;
protected synchronized java.util.Map getPort2NamespaceMap() {
if (port2NamespaceMap == null) {
port2NamespaceMap = new java.util.HashMap();
port2NamespaceMap.put(
"TestService",
"http://schemas.xmlsoap.org/wsdl/soap/");
}
return port2NamespaceMap;
}
```

getPort2NamespaceMap() creates a map that uses the WSDL port as a key and namespace as a value. It returns this map. Recall that getTestService() invokes a get() on this returned map, passing in testServicePortName as a key to get the namespace value:

```
...(String) getPort2NamespaceMap().get(testServicePortName)...
```

But remember that testServicePortName resolves to "Test-Service", yet the getPort2NamespaceMap() method never added a mapping using "Test-Service" as a key. It used the string "TestService" instead:

```
port2NamespaceMap.put(
"TestService",
"http://schemas.xmlsoap.org/wsdl/soap/");
```

Therefore, the returned namespace value is null, which is why the Web services engine throws the `javax.xml.rpc.ServiceException: WSWS5055E: Cannot find a Service for namespace null` exception.

The fix for the APAR corrects WSDL2Java so that it generates the *wsdl_service*Locator class to use the XML QName local part of the WSDL port.

3.4.6 Serialized SOAP message does not contain children in body

This problem occurs when a document/literal WSDL defines a message without a part. At run time, the serialized SOAP message does not contain any children under the SOAP Body.

Consider a document/literal WSDL with the following message defined:

```
<wsdl:message name="testHelloRequest"/>
```

This message is used as input for the operation testHello shown in Example 3-30.

Example 3-30 Input message in WSDL operation

```
<wsdl:operation name="testHello">
<wsdl:input name="testHelloRequest" message="intf:testHelloRequest"/>
…
</wsdl:operation>
```

A Web service client developed from such a WSDL and running in the WebSphere environment would send a message like that shown in Example 3-31.

Example 3-31 Message with empty SOAP Body

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body/>
</soapenv:Envelope>
```

Notice that the SOAP Body element is empty.

Resolve the problem

If the SOAP Body was not intended to be empty, correct the WSDL, and re-develop and re-deploy the application.

More about this problem

The WSDL defines a document/literal message, as specified in the binding. See Example 3-32.

Example 3-32 WSDL defining a document/literal message

```
<wsdl:binding name="TestSoapBinding" type="intf:Test">
    <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="testHello">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="testHelloRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="testHelloResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="helloWorldException">
        <wsdlsoap:fault name="helloWorldException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
```

Also remember the input message, testHelloRequest? It does not contain a part:

```
<wsdl:message name="testHelloRequest"/>
```

Basic Profile Version 1.1: The Basic Profile Version 1.1 says this on the matter of child elements in document/literal messages:

> 4.7.8 Child Element for Document-Literal Bindings
>
> WSDL 1.1 is not completely clear what, in document-literal style bindings, the child element of soap:Body is.
>
> R2712 A document-literal binding MUST be serialized as an ENVELOPE with a soap:Body whose child element is an instance of the global element declaration referenced by the corresponding wsdl:message part.

In order for the request SOAP Body to contain any elements, the input message must contain a part that references an element, globally defined in the schema. See Example 3-33.

Example 3-33 Input message, now with a part

```
<wsdl:message name="testHelloRequest">
    <wsdl:part name="parameters" element="impl:testHello"/>
  </wsdl:message>
```

The testHello element is declared as in Example 3-34.

Example 3-34 testHello element

```
<element name="testHello">
    <complexType>
     <sequence>
      <element name="name" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
```

Remember that elements can be declared using anonymous complexTypes. The complexType definition specifies that element testHello contains a child element, called name.

It is also important to understand that changing the WSDL often necessitates changing the Java code — including development and deployment code. In this particular example, a change to the input message changed the signature of the mapped Java method, testHello(), and a slight change in the implementation in both client and service applications. Web services deployment code also had to be re-generated.

Example 3-35 shows the ensuing SOAP request message after the applications were re-built and re-deployed.

Example 3-35 SOAP request message with SOAP Body containing child elements

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
      <p585:testHello xmlns:p585="http://test.com">
         <name>Wendy</name>
      </p585:testHello>
   </soapenv:Body>
</soapenv:Envelope>
```

In this example, the document/literal request message has an empty SOAP Body because the input message has no part. However, this cause and effect pattern is not limited to just the request, but also to the response. In other words, if the output message similarly contains no part, then the response message's SOAP Body will also not contain any elements.

Support preparation

If changing the WSDL and the application does not resolve the problem, collect the following data for use by IBM support:

- Complete test case, including WSDL and Java source
- Web services trace that reflects the sending of the message with the empty SOAP body

Go to “The next step” on page 155 for information about performing an online search of Web services problems and for a list of useful technical articles. If these resources do not help you resolve the problem, contact IBM support.

3.4.7 Improperly formed SOAP request or response

The WSWS3047E deserialization error is a common problem. It usually occurs in the case where either a WebSphere Web service requestor or provider is interoperating with a requestor or provider on another Web services platform. The incoming SOAP message, whether request or response, is not formed properly. The error does not happen because of a product defect.

Resolve the problem

Fix the message so that local elements are qualified or unqualified and in the proper namespace according to the schema. You should work with the Web services vendor to accomplish this task.

More about this problem

Consider the case where a non-WebSphere client sends a request to a WebSphere Web service. Examine the request message (Example 3-36) that resulted in the WebSphere Web service sending a SOAP fault as a response.

Example 3-36 Request message

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
      <p585:testHello xmlns:p585="http://test.com">
         <p585:name>Wendy</p585:name>
      </p585:testHello>
   </soapenv:Body>
</soapenv:Envelope>
```

This request is sent to the target Web service testHello, running on WebSphere.

The testHello Web service operation is defined in the WSDL as shown in Example 3-37.

Example 3-37 testHello Web service operation

```
<wsdl:operation name="testHello">
      <wsdl:input name="testHelloRequest"
message="impl:testHelloRequest"/>
      <wsdl:output name="testHelloResponse"
message="impl:testHelloResponse"/>
    </wsdl:operation>
```

In this example, since the deserialization problem occurs during processing of request, the input message is of interest. See Example 3-38.

Example 3-38 Input message

```
<wsdl:message name="testHelloRequest">
    <wsdl:part name="parameters" element="impl:testHello"/>
  </wsdl:message>
```

Now look at the element that the message part refers to. `testHello` is declared in the WSDL's embedded schema. See Example 3-39.

Example 3-39 testHello global element

```
<schema targetNamespace="http://test.com"
 xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <element name="testHello">
    <complexType>
     <sequence>
      <element name="name" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
  </schema>
```

testHello is a global element. It is declared immediately under the schema element, and not within a complexType definition. Therefore, it *exists* in that schema's target namespace, http://test.com. The testHello element appears correctly, according to the schema it is declared in:

```
...<p585:testHello xmlns:p585="http://test.com">...
```

Notice that testHello is prefixed with the string p585 and that p585 is set to namespace http://test.com.

Notice in the testHello element's anonymous complexType definition that the child element, name, is declared. Remember that according to the error message, the Web services engine complains that it cannot deserialize element name. Example 3-40 shows how the name element appears in the SOAP request.

Example 3-40   testHello element in SOAP request

```
…
      <p585:testHello xmlns:p585="http://test.com">
         <p585:name>Wendy</p585:name>
      </p585:testHello>
      ...
```

Notice that name is qualified, too, with prefix p585 that resolves to namespace http://test.com.

Note: The W3C recommendation, "XML Schema Part 0: Primer Second Edition," says the following about qualifying local elements:

> Qualification of local elements and attributes can be globally specified by a pair of attributes, elementFormDefault and attributeFormDefault, on the schema element, or can be specified separately for each local declaration using the form attribute. All such attributes' values may each be set to unqualified or qualified, to indicate whether or not locally declared elements and attributes must be unqualified.

In this case, name is a local element because it is not declared within the context of the schema — it is declared within a complexType definition. Because the schema in which `name` is declared does not explicitly have the elementFormDefault attribute, the default of *unqualified* is assumed. Therefore, name should not be qualified in the SOAP message.

Example 3-41 shows how you should not specify name (qualified with p585).

Example 3-41 Incorrect specification of name

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
      <p585:testHello xmlns:p585="http://test.com">
         <p585:name>Wendy</p585:name>
      </p585:testHello>
   </soapenv:Body>
</soapenv:Envelope>
```

Example 3-42 shows the correct way to specify name (unqualified).

Example 3-42 Correct specification of name

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
      <p585:testHello xmlns:p585="http://test.com">
         <name>Wendy</name>
      </p585:testHello>
   </soapenv:Body>
</soapenv:Envelope>
```

Alternatively, the use of a default namespace (Example 3-43) is another way that the message can be structured so that name is in the wrong namespace.

Example 3-43 Incorrect specification using the default namespace

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
      <testHello xmlns="http://test.com">
         <name>Wendy</name>
      </testHello>
   </soapenv:Body>
</soapenv:Envelope>
```

The appearance of the xmlns= attribute on parent element testHello indicates that the default namespace is http://test.com. Therefore, its child element name inherits this default namespace. But again, according to the schema, name does not have a namespace.

Instead, the message should appear as shown in Example 3-44 to be valid.

Example 3-44 Correct specification using the default namespace

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
      <testHello xmlns="http://test.com">
         <name xmlns="">Wendy</name>
      </testHello>
   </soapenv:Body>
</soapenv:Envelope>
```

xmlns="" indicates that name exists in no default namespace at all.

The only way to resolve the WSWS3047E error is to fix the message so that local elements are qualified or unqualified, according to the schema. Additionally, whether elements are qualified using a prefix or default namespace, the namespace must be the correct one. You should work with the Web services vendor to accomplish this task.

Support preparation

If you still observe the WSWS3047E error after fixing the message, collect the following data for use by IBM support:

- Capture of SOAP request and response. These can be obtained using the TCPMonitor tool. (See “Collecting trace data with TCPMonitor” on page 153.)
- Web services engine trace reflecting the "`org.xml.sax.SAXException: WSWS3047E: Error: Cannot deserialize element <`*`element_name`*`> of bean <`*`bean_name`*`>`" error. See “Enabling the Web services trace” on page 154 for information about collecting the trace.
- Complete test case. If you cannot provide the test case, then the WSDLs with all referenced schema files are needed.

Go to “The next step” on page 155 for information about performing an online search of Web services problems and for a list of useful technical articles. If these resources do not help you resolve the problem, contact IBM support.

3.5 Collecting diagnostic data

This section provides information for collecting diagnostic data useful in diagnosing Web services problems.

3.5.1 Collecting JVM logs

JVM logs, often referred to as SystemOut and SystemErr logs, are created for every WebSphere Application Server process (application server, cluster member, node agent, and deployment manager). They can be found in the following locations:

- WebSphere Application Server V6.x for z/OS

  The JVM logs are located in the address space output. Generally, the SYSPRINT card correlates to SystemOut.log, and SYSOUT correlates to SystemErr.log.

- WebSphere Application Server V6.x (distributed and i5/OS)

  The JVM log files are by default named SystemOut.log and SystemErr.log. The default location for the SystemOut and SystemErr logs is:

  – *profile_root*/logs/*server_name*/SystemOut.log
  – *profile_root*/logs/*server_name*/SystemErr.log

The location of application server logs is configurable.

a. Select **Troubleshooting** → **Logs and Trace** in the navigation bar.
b. Click the server name.
c. Select **JVM logs**.

This page shows the location of the log file.

3.5.2 Collecting trace data with TCPMonitor

You can use the TCPMonitor tool to observe and capture SOAP messages exchanged between a Web service client and Web service provider. TCPMonitor redirects messages from a port, records the messages, and forwards the messages to another port.

1. In the WebSphere bin directory, create a tcpmon script specific for your platform (a batch script for Windows and a shell script for UNIX).
   - For a Windows script, the file would contain the following:
     ```
     call "setupCmdLine.bat"
     %JAVA_HOME%\bin\java
     -Djava.ext.dirs=%WAS_EXT_DIRS%;%WAS_HOME%\plugins
     com.ibm.ws.webservices.engine.utils.tcpmon
     ```
   - On a UNIX system, the file would look like the following:
     ```
     . ./setupCmdLine.sh
     ${JAVA_HOME}/bin/java
     -Djava.ext.dirs=${WAS_EXT_DIRS}:${WAS_HOME}/plugins
     com.ibm.ws.webservices.engine.utils.tcpmon
     ```
2. Run the script in a command-line console. A GUI should appear.
3. In the GUI, specify the following:
   - Listen Port

     Port that you want TCPMonitor to listen on. This port name should be unused.
   - Target Hostname

     Hostname of Web service provider.
   - Target Port

     Port of the Web service provider.
4. Click **Add**.

   TCPMonitor is now ready to intercept SOAP requests and responses.

   The Web service client should now redirect its request through TCPMonitor. The new endpoint URL given to the client should contain the host name where TCPMonitor is running and the listen port specified in the TCPMonitor Listen Port text field.

For example, consider the following case:

– The Web service provider is accessible at endpoint http://hostA:9080/Hello/HelloWorld.
– TCPMonitor is running on hostB and is listening on port 9085.

To redirect the client request through TCPMonitor, point the client to endpoint http://hostB:9085/Hello/HelloWorld.

5. Invoke the client. You should observe both the SOAP request and response in the two panels of the GUI. The messages can be saved to a text file by clicking the **Save** button.

3.5.3 Enabling the Web services trace

You can enable and gather Web services runtime trace for an unmanaged client, a managed client running in the J2EE client container, managed clients running on the Web or EJB containers, or Web service providers running on the Web or EJB containers.

Enabling and gathering trace for an unmanaged client

To enable and gather trace for an unmanaged client, do the following:

1. Create a trace properties file by copying the *app_server_root*\properties\TraceSettings.properties file to the same directory as your client application Java archive (JAR) file.
2. Edit the properties file and change the traceFileName value to the location for the trace data output (for example, traceFileName=c:\\temp\\myAppClient.trc).
3. Edit the properties file to remove com.ibm.ejs.ras.*=all=enabled and add `com.ibm.ws.webservices.*=all=enabled`.
4. Add the following options to the Java command line that is used to run the client:

```
-DtraceSettingsFile=trace_properties_file
-Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
-Djava.util.logging.configureByServer=true
```

Where *trace_properties_file* represents the name of the properties file that you created in the previous steps. For example:

```
java -DtraceSettingsFile=TraceSettings.properties
-Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
-Djava.util.logging.configureByServer=true myApp.myAppMainClass
```

Enabling and gathering trace data for a managed client

To enable and gather trace for a managed client running in the J2EE client container, invoke the launchClient command-line tool with the following options:

```
-CCtrace=com.ibm.ws.webservices.*=all=enabled
-CCtracefile=traceFileName
```

For example:

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

Enabling the trace from the administrative console

To enable and gather trace for managed clients running in the Web or EJB container, or Web service providers running in the Web or EJB containers, do the following:

1. In the administrative console, expand **Servers** → **Application Servers** → *server_name*.
2. Select **Diagnostic Trace Service**.
3. Set Maximum File Size to 200 MB and select the **File radio** button in the General properties section. Set the appropriate number of historical trace files.
4. Navigate to **Change Log Detail Levels** in the Additional properties section.
5. Clear the current trace string and add the following trace string to the General properties field for general Web services issues:

   ```
   *=info:com.ibm.ws.webservices.*=all
   ```

By default, the trace is logged to a file called trace.log in the same location as SystemOut.log and SystemErr.log, *profile_root/*logs/*server_name.*

Disabling the trace

To restore trace state back to normal, use this same process and clear the trace string. Restart the server.

3.6 The next step

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, there are other things that can go wrong.

3.6.1 Search online support

If you are sure that the problem is a result of developing, deploying, or executing a Web services application, there are tasks that you can do before contacting IBM support. First, you should review the documentation that you have gathered for errors that were not addressed in this paper and search support sites for information or fixes. Look for current information available from IBM support on known issues and resolutions on the following IBM support page:

http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCR4XC

3.6.2 Review troubleshooting articles

Look also at the WebSphere Information Center documentation for additional resources for diagnosing and fixing issues with Web services applications. The following list provides the URL for the distributed platform Network Deployment topic. If you are using a different package or platform, you can find the comparable topic by using the title (enclosed in double quotation marks) as a search argument.

- *Troubleshooting Web Services*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/twbs_troubleshootwbs.html

  This topic includes a number of links to sub-topics, both within the text and at the bottom, to specific diagnostic information for different types of Web services problems.

- *Web services serialization and deserialization troubleshooting tips*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rwbs_trbserialize.htm

- *Trace and logging for WSIF*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/twsf_tracelog.html

- *UDDI registry troubleshooting*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/twsu_probdet.html

- *Universal Discovery, Description, and Integration, Web Service, and SOAP component troubleshooting tips*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rtrb_svsccomp.html

- Explanation of messages

  The WebSphere information center has a message reference that provides message text and user response suggestions for each message prefix type.

  The information center is at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

  Navigate to ***WebSphere package*** → **Reference** → **Messages** → ***message_prefix***.

3.6.3 Resources for learning

For general information about Web services applications for WebSphere Application Server, see:

- *Web services* in the WebSphere Information Center, including links to tutorials, samples, and resources for learning:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc6tech_wbs.html

- *Web Services Handbook for WebSphere Application Server 6.1*

  http://www.redbooks.ibm.com/abstracts/sg247257.html

Understanding the following specifications is important when diagnosing problems with Web services applications:

- The JAX-RPC 1.1 specification

  http://developers.sun.com/techtopics/webservices/reference/api/index.html

- The Web services for J2EE specification (JSR 109)

  http://jcp.org/en/jsr/detail?id=109

- Basic Profile Version 1.1

  http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

  This specification describes how Web services should behave in order to interoperate with each other. It is key in solving problems that arise from scenarios where the client and provider are on different Web services vendors (one on WebSphere and another on .Net, for example).

- SOAP 1.1

  http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

  This specification describes how SOAP 1.1 messages should appear.

- SOAP with Attachments API for Java (SAAJ) Specification 1.1

  `https://saaj.dev.java.net/`

  SAAJ is an alternative programming model to JAX-RPC.
- Web Services Description Language (WSDL) 1.1

  `http://www.w3.org/TR/wsdl`

  This outlines how a compliant WSDL should be structured.
- XML Schema

  `http://www.w3.org/2001/XMLSchema`

3.6.4 Contact IBM

If these steps do not resolve your problem, gather additional information and raise a problem record with IBM. If instructions on what to gather were not included for your particular problem area, use the MustGather information for your problem to collect documentation: MustGather: Web services engine and tooling problems for WebSphere Application Server V6.1, V6, V5.1 and V5.

`http://www-1.ibm.com/support/docview.wss?rs=180&context=SSCR4XC&q1=MustGatherDocument&uid=swg21198363&loc=en_US&cs=utf-8&lang=en`

The following URL contains a list of all of the MustGather documentation for Web services problems involving the Web services engine, WSIF, or Web services gateway:

`http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCR4XC&q=MustGatherDocument`

The following URL contains a list of the MustGather documentation for WS-Security:

`http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCNPY4&q=MustGatherDocument`

Part 4

Class loader problem determination

This part contains the following IBM Redpaper:

- *WebSphere Application Server V6.1: Classloader Problem Determination*, REDP-4307 by Jasper Chui

4

Class loader problem determination

Troubleshooting class loader issues with applications running on WebSphere Application Server version 6.1 requires a basic understanding of how class loaders work within WebSphere.

This paper begins with an introduction on understanding class loaders and class loaders within WebSphere Application Server. This includes an explanation of configuration options for the class loader and how they can be used with your application to achieve a specific behavior. Next we provide information that will help you recognize when a class loader problem occurs and propose some solutions to help you resolve them. This process includes instructions on collecting class loader specific traces, identifying and gathering exception specific trace information, analyzing the traces, and an explanation of possible root causes and solutions to these common class loader exceptions.

A sample application, ClassloaderSampleEAR.zip, is available for download with this paper. This sample can be used to reproduce some of the common class loader exceptions visited in this document. We encourage you to review it and to follow the analysis described in this document to help you better understand how to recognize and how to resolve class loader issues within WebSphere Application Server V6.1. The instructions to install and to use this application are in “Installing and configuring the sample application” on page 214.

4.1 Introduction to class loaders

Understanding how the Java and WebSphere class loaders work is critical to packaging and deploying J2EE applications. Failure to set up the class loaders properly most likely results in a cascade of the infamous class loading exceptions (such as ClassNotFoundException) when trying to start your application.

This paper explains class loaders and how to customize the behavior of the WebSphere class loaders to suit your particular application's requirements. It includes the following:

- Brief introduction to Java class loaders
- WebSphere class loaders overview
- Configuring WebSphere for class loaders

4.1.1 Brief introduction to Java class loaders

Class loaders enable the Java virtual machine (JVM) to load classes. Given the name of a class, the class loader locates the definition of this class. Each Java class must be loaded by a class loader.

When you start a JVM, you use three class loaders: the bootstrap class loader, the extensions class loader, and the application class loader.

The bootstrap class loader is responsible for loading only the core Java libraries, which are vm.jar, core.jar, and so on, in the *Java_home*/jre/lib directory, where *Java_home* is the installation directory for the JDK. This class loader, which is part of the core JVM, is written in native code.

The extensions class loader is responsible for loading the code in the extensions directories (*Java_home*/jre/lib/ext or any other directory specified by the java.ext.dirs system property). This class loader is implemented by the sun.misc.Launcher$ExtClassLoader class.

The application class loader is responsible for loading the code that is found on java.class.path, which ultimately maps to the system CLASSPATH variable. This class loader is implemented by the sun.misc.Launcher$AppClassLoader class.

The parent-delegation model is a key concept to understand when dealing with class loaders. It states that a class loader delegates class loading to its parent before trying to load the class itself. The parent class loader can be either another custom class loader or the bootstrap class loader. But what is very important is that a class loader can only delegate requests to its parent class loader, never to its child class loaders (it can go up the hierarchy but never down).

The extensions class loader is the parent for the application class loader. The bootstrap class loader is the parent for the extensions class loader. The class loaders hierarchy is shown in Figure 4-1.

If the application class loader needs to load a class, it first delegates to the extensions class loader, which, in turn, delegates to the bootstrap class loader. If the parent class loader cannot load the class, the child class loader tries to find the class in its own repository. In this manner, a class loader is only responsible for loading classes that its ancestors cannot load.

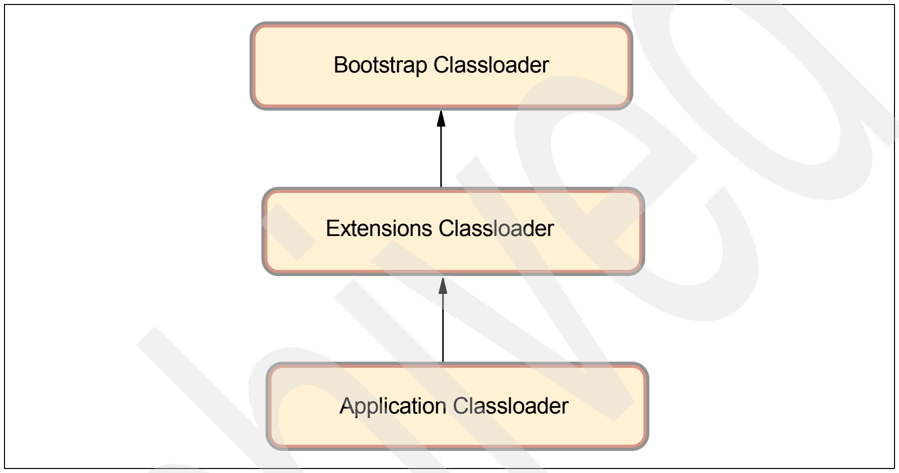


Figure 4-1 Java class loaders hierarchy

4.1.2 WebSphere class loaders overview

Note: Keep in mind when reading the following discussion that each JVM has its own setup of class loaders. In a WebSphere environment hosting multiple application servers (JVMs), this means the class loaders for the JVMs are completely separated even if they are running on the same physical machine.

Also note that the Java Virtual Machine (JVM) uses class loaders called the extensions and application class loaders. As you will see, the WebSphere run time also uses class loaders called extensions and application class loader, but despite their names they are not the same as the JVM ones.

WebSphere provides several custom delegated class loaders, as shown in Figure 4-2 on page 164.

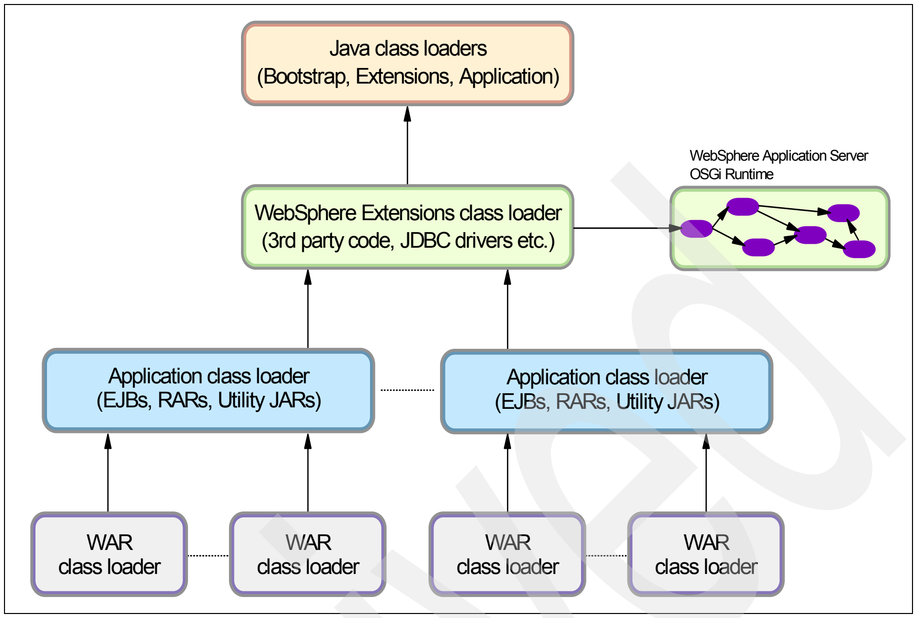

Java class loaders
(Bootstrap, Extensions, Application)
WebSphere Application Server
OSGi Runtime
WebSphere Extensions class loader
(3rd party code, JDBC drivers etc.)
Application class loader
(EJBs, RARs, Utility JARs)
Application class loader
(EJBs, RARs, Utility JARs)
WAR
class loader
WAR
class loader
WAR
class loader
WAR
class loader

Figure 4-2 WebSphere class loaders hierarchy

The top box represents the Java (bootstrap, extensions, and application) class loaders. WebSphere loads just enough here to get itself bootstrapped and to initialize the WebSphere extensions class loader.

4.1.3 WebSphere extensions class loader

New in V6.1: The WebSphere extensions class loader is where WebSphere itself is loaded. In previous versions of WebSphere, the run time was loaded by this single class loader. However, beginning with WebSphere Application Server V6.1, WebSphere is now packaged as a set of OSGi bundles. Each OSGi bundle is loaded separately by its own class loader. This network of OSGi class loaders is then connected to the extensions class loader and the rest of the class loader hierarchy through an OSGi gateway class loader.

Despite this architectural change in the internals of how WebSphere loads its own classes, there is no behavioral change as far as your applications are concerned. They still have the same visibility, and the same class loading options still exist for your application.

New in V6.1: In previous versions of WebSphere Application Server the WebSphere run time class files were stored in the classes, lib, lib/ext, and installedChannels directories in the *app_server_root* directory. Because of the OSGi packaging, these directories no longer exist and the run time classes are now stored in the *app_server_root*/plugins directory.

The class path used by the extensions class loader is retrieved from the ws.ext.dirs system property, which is initially derived from the WAS_EXT_DIRS environment variable set in the setupCmdLine script file. The default value of ws.ext.dirs is displayed in Example 4-1 .

Example 4-1 Default value of we.ext.dirs

```
SET
WAS_EXT_DIRS=%JAVA_HOME%\lib;%WAS_HOME%\classes;%WAS_HOME%\lib;%WAS_HOME%\insta
lledChannels;%WAS_HOME%\lib\ext;%WAS_HOME%\web\help;%ITP_LOC%\plugins\com.ibm.e
tools.ejbdeploy\runtime
```

Each directory listed in the ws.ext.dirs environment variable is added to the WebSphere extensions class loaders class path, and every JAR file and ZIP file in the directory is added to the class path.

As you can see, even though the classes and installedChannels directories no longer exist in the *app_server_root* directory for V6.1, the setupCmdLine script still adds them to the extensions class path. This means that if you previously added your own JAR files to, for example, the *app_server_root*/lib directory, you could create this directory, and add your JAR files to it and they would still be loaded by the extensions class loader. However, this is not recommended and you should really try to migrate away from such a setup.

On the other hand, if you developed Java applications that rely on the WebSphere JAR files that were previously in the *app_server_root*/lib directory, you will need to modify your application to retain compatibility. WebSphere Application Server V6.1 provides two thin client libraries designed specifically for such applications: one administrative client library and one Web services client library. These thin client libraries can be found in the following *app_server_root*/runtimes directories:

- com.ibm.ws.admin.client_6.1.0.jar
- com.ibm.ws.webservices.thinclient_6.1.0.jar

These libraries provide everything your application might need for connecting to and working with WebSphere.

New in V6.1: WebSphere Application Server V6.1 gives you the ability to restrict access to internal WebSphere classes so that your applications do not make unsupported calls to WebSphere classes not published in the official WebSphere Application Server API. This setting is a per-server (JVM) setting called Access to internal server classes.

The default setting is Allow, meaning that your applications can make unrestricted calls to non-public internal WebSphere classes. This is not recommended and may be prohibited in future releases. Therefore, as an administrator, it is a good idea to switch this setting to *Restrict* to see if your applications still work. If they depend on non-public WebSphere internal classes, you will receive a ClassNotFoundException, and in that case you can switch back to Allow. Your developers should then try to migrate their applications so that they do not make unsupported calls to the WebSphere internal classes in order to retain compatibility with future WebSphere Application Server releases.

4.1.4 Application and Web module class loaders

J2EE applications consist of five primary elements: Web modules, EJB modules, application client modules, resource adapters (RAR files), and utility JARs. Utility JARs contain code that can be used by both EJBs and servlets. Utility frameworks such as log4j are good examples of a utility JAR.

EJB modules, utility JARs, resource adapter files, and shared libraries associated with an application are always grouped together into the same class loader. This class loader is called the application class loader. Depending on the class loader policy, this class loader can be shared by multiple applications (EARs) or be unique for each application, which is the default.

By default, Web modules receive their own class loader, a WAR class loader, to load the contents of the WEB-INF/classes and WEB-INF/lib directories. You can modify the default behavior by changing the application's WAR class loader policy. The default is to have a class loader for each WAR file in the application (this setting was called Module in previous releases). If the WAR class loader policy is set to Single class loader for application (called Application in previous releases), the Web module contents are loaded by the application class loader in addition to the EJBs, RARs, utility JARs, and shared libraries. The application class loader is the parent of the WAR class loader.

The application and the WAR class loaders are reloadable class loaders. They monitor changes in the application code to automatically reload modified classes. You can modify this behavior at deployment time.

4.1.5 Handling JNI code

Because a JVM only has a single address space and native code can only be loaded once per address space, the JVM specification states that native code may only be loaded by one class loader in a JVM.

This may cause a problem if, for example, you have an application (EAR file) with two Web modules that both need to load the same native code through a Java Native Interface (JNI™). Only the Web module that first loads the library will succeed.

To solve this problem, you can break out just the few lines of Java code that load the native code into a class on its own and place this file on WebSphere's application class loader (in a utility JAR). However, if you would deploy multiple such applications (EAR files) to the same application server (JVM), you would have to place the class file on the WebSphere extensions class loader instead to ensure the native code is only loaded once per JVM.

If the native code is placed on a reloadable class loader (such as the application class loader or the WAR class loader), it is important that the native code can properly unload itself should the Java code have to reload. WebSphere has no control over the native code and if it does not unload and load properly the application may fail.

If one native library depends on another one, things become even more complicated. Search for Dependent native library in the Information Center for more details.

4.1.6 Configuring WebSphere for class loaders

In the previous topic, you learned about WebSphere class loaders and how they work together to load classes. There are settings in WebSphere Application Server that allow you to influence WebSphere class loader behavior. This section discusses these options.

4.1.7 Class loader policies

For each application server in the system, the class loader policy can be set to Single or Multiple.

When the application server class loader policy is set to Single, a single application class loader is used to load all EJBs, utility JARs, and shared libraries within the application server (JVM). If the WAR class loader policy then was set to Single class loader for application (or Application), then the Web module contents for this particular application are also loaded by this single class loader.

When the application server class loader policy is set to Multiple, which is the default, each application will receive its own class loader for loading EJBs, utility JARs, and shared libraries. Depending on whether the WAR class loader loading policy is set to class loader for each WAR file in application (or Module) or Single class loader for application (or Application), the Web module might or might not receive its own class loader.

Following is an example to illustrate. You have two applications, Application1 and Application2, running in the same application server. Each application has one EJB module, one utility JAR, and two Web modules. If the application server has its class loader policy set to Multiple, the default, and the class loader policy for all the Web modules are set to use a class loader for each WAR file in application (Module), also the default, the result is as shown in Figure 4-3.

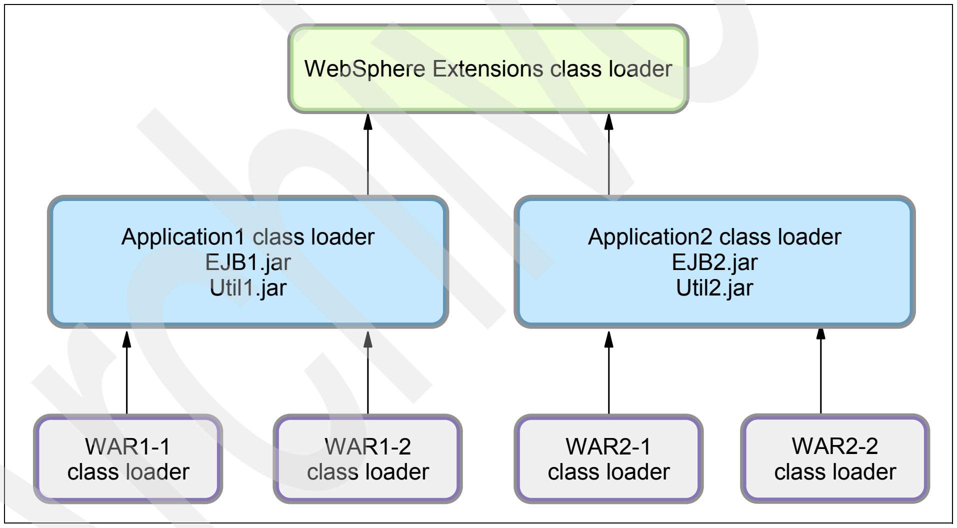


Figure 4-3   Class loader policies: Example 1

Each application is completely separated from the other, and each Web module is also completely separated from the other one in the same application. WebSphere's default class loader policies result in total isolation between the applications and the modules.

If we now change the class loader policy for the WAR2-2 module from class loader for each WAR file in application (Module) to Single class loader for application (Application), the result is shown in Figure 4-4.

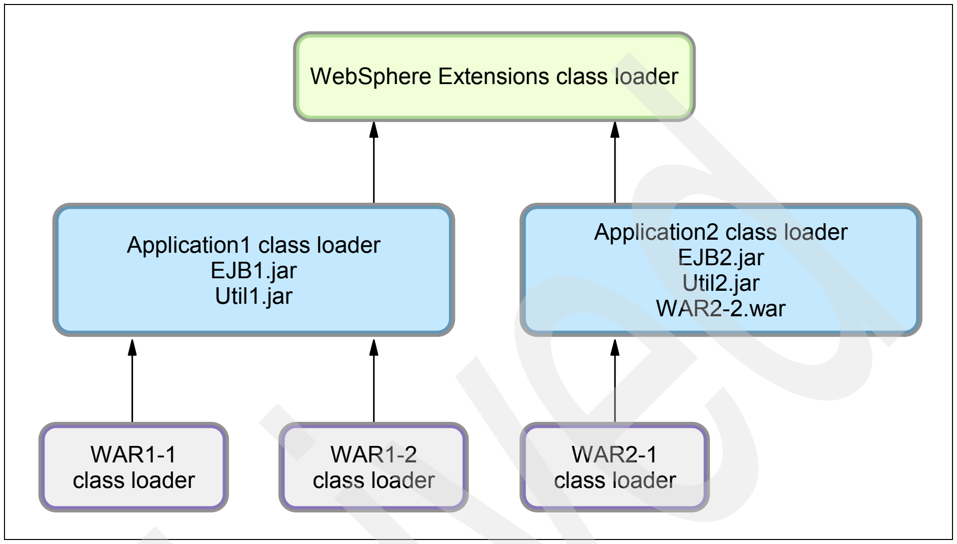


Figure 4-4 Class loader policies: Example 2

Web module WAR2-2 is loaded by Application2’s class loader and classes, for example, in Util2.jar, we can see classes in WAR2-2’s /WEB-INF/classes and /WEB-INF/lib directories.

As a last example, if we change the class loader policy for the application server from Multiple to Single and also change the class loader policy for WAR2-1 from class loader for each WAR file in application (Module) to Single class loader for application (Application), the result is as shown in Figure 4-5 on page 170.

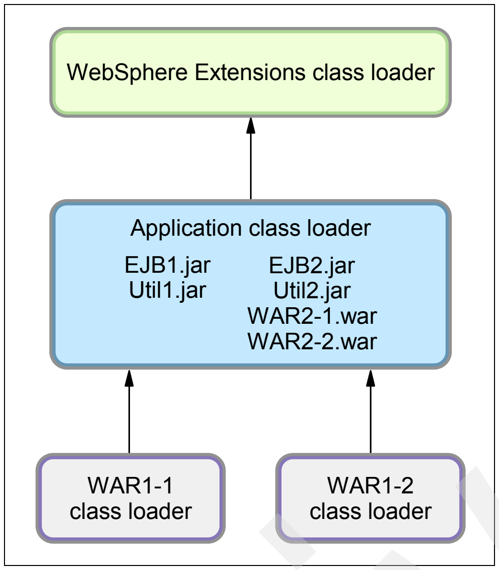


Figure 4-5 Class loader policies: Example 3

There is now only a single application class loader loading classes for both Application1 and Application2. Classes in Util1.jar can see classes in EJB2.jar, Util2.jar, WAR2-1.war and WAR2-2.war. The classes loaded by the application class loader still cannot, however, see the classes in the WAR1-1 and WAR1-2 modules, because a class loader can only find classes by going up the hierarchy, never down.

4.1.8 Class loading/delegation mode

WebSphere's application class loader and WAR class loader both have a setting called the class loader order. This setting determines whether they should follow the normal Java class loader delegation mechanism or override it.

There are two possible options for the class loading mode:

- Classes loaded with parent class loader first
- Classes loaded with application class loader first

In previous WebSphere releases, these settings were called PARENT_FIRST and PARENT_LAST, respectively.

The default value for class loading mode is Classes loaded with parent class loader first (PARENT_FIRST). This mode causes the class loader to first delegate the loading of classes to its parent class loader before attempting to load the class from its local class path. This is the default policy for standard Java class loaders.

If the class loading policy is set to Classes loaded with application class loader first (PARENT_LAST), the class loader attempts to load classes from its local class path before delegating the class loading to its parent. This policy allows an application class loader to override and provide its own version of a class that exists in the parent class loader.

Note: The administrative console is a bit confusing at this point. On the settings page for a Web module, the two options for class loader order are Classes loaded with parent class loader first and Classes loaded with application class loader first. However, in this context, the "application class loader" really refers to the WAR class loader, so the option Classes loaded with application class loader first should really be called Classes loaded with WAR class loader first.

Assume you have an application, similar to Application1 in the previous examples, and it uses the popular log4j package to perform logging from both the EJB module and the two Web modules. Also assume that each module has its own, unique, log4j.properties file packaged into the module. It is tempting to configure log4j as a utility JAR so you only have a single copy of it in your EAR file.

However, if you do that, you might be surprised to see that all modules, including the Web modules, load the log4j.properties file from the EJB module. The reason is that when a Web module initializes the log4j package, the log4j classes are loaded by the application class loader. log4j is configured as a utility JAR. log4j then looks for a log4j.properties file on its class path and finds it in the EJB module.

Even if you do not use log4j for logging from the EJB module and the EJB module does not, therefore, contain a log4j.properties file, log4j does not find the log4j.properties file in any of the Web modules anyway. The reason is that a class loader can only find classes by going up the hierarchy, never down.

To solve this problem, you can do one of the following:

- Create a separate file, for example, Resource.jar, configure it as a utility JAR, move all log4j.properties files from the modules into this file, and make their names unique (like war1-1_log4j.properties, war1-2_log4j.properties, and ejb1_log4j.properties). When initializing log4j from each module, tell it to load the proper configuration file for the module instead of the default (log4j.properties).

- Keep the log4j.properties for the Web modules in their original place (/WEB-INF/classes), add log4j.jar to both Web modules (/WEB-INF/lib) and set the class loading mode for the Web modules to Classes loaded with application class loader first (PARENT_LAST). When initializing log4j from a Web module, it loads the log4j.jar from the module itself and log4j would find the log4j.properties on its local classpath, the Web module itself. When the EJB module initializes log4j, it loads from the application class loader and it finds the log4j.properties file on the same class path, the one in the EJB1.jar file.
- Merge, if possible, all log4j.properties files into one, and place it on the application class loader, in a Resource.jar file, for example.

Singletons: The Singleton pattern is used to ensure that a class is instantiated only once. However, *once* only means *once for each class loader*. If you have a Singleton being instantiated in two separated Web modules, two separate instances of this class are created, one for each WAR class loader. So in a multi-class loader environment, special care must be taken when implementing Singletons.

4.1.9 Shared libraries

Shared libraries are files used by multiple applications. Examples of shared libraries are commonly used frameworks like Apache Struts or log4j. You use shared libraries typically to point to a set of JARs and associate those JARs to an application, a Web module, or the class loader of an application server. Shared libraries are especially useful when you have different versions of the same framework you want to associate to different applications.

Shared libraries are defined using the administration tools. They consist of a symbolic name, a Java class path, and an optional native path for loading JNI libraries. They can be defined at the cell, node, server, or cluster level. However, simply defining a library does not cause the library to be loaded. You must associate the library to an application, a Web module, or the class loader of an application server for the classes represented by the shared library to be loaded. Associating the library to the class loader of an application server makes the library available to all applications on the server.

Note: If you associate a shared library to an application, do not associate the same library to the class loader of an application server.

Associate a shared library with an application

You can associate the shared library to an application in one of two ways:

- You can use the administration tools. For information about using this method, see “Associate the shared library with an application” on page 199.
- You can use the manifest file of the application and the shared library. The shared library contains a manifest file that identifies it as an extension. The dependency to the library is declared in the application’s manifest file by listing the library extension name in an extension list.

  For more information about this method, search for installed optional packages in the Information Center.

Associate a shared library with a server class loader

Shared files are associated with the class loader of an application server using the administrative tools.

The settings are found in the Server Infrastructure section.

1. Expand the **Java and Process Management**.
2. Select **Class loader**, and then click the **New** button to define a new class loader.
3. After you have defined a new class loader, you can modify it and, using the **Shared library references** link, you can associate it to the shared libraries you need.

4.2 Problem determination for class loader exceptions

This section takes you through the process of identifying and resolving common class loader problems. It helps you identify when a class loader problem occurs and helps you to collect class loader specific traces and other diagnostic data. It will help you analyze the diagnostic data and identify possible root causes of the problem and the solution for those root causes.

Note: The examples in this chapter were created using the sample application, ClassloaderSampleEAR, available for download with this paper (see “Installing and configuring the sample application” on page 214).

4.2.1 Identify symptoms

Problems related to class loaders are normally identified during the course of diagnosing general application errors. Depending on how the application handles unexpected exceptions, this can look like any other type of application failure (HTTP 404 Page cannot be displayed, for example).

Tip: A common example of a class loader problem occurring is when **log4j** is not producing logs. The root cause is that the class loader is loading the WebSphere log properties files instead of your own properties files.

For more information about using custom logging, see the following Web site:

- Jakarta Commons Logging

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/ctrb_classload_jcl.html

4.2.2 Collect SystemOut

Indications of class loader issues are not readily apparent until you view the JVM SystemOut log. Class loader problems normally appear as an exception and are accompanied with a Java stack trace.

Determine the type of error

Typical class loader exceptions include the following:

- java.lang.ClassCastException
- java.lang.ClassNotFoundException
- java.lang.NoClassDefFoundError
- java.lang.NoSuchMethodError, java.lang.IllegalArgumentException
- java.lang.UnsatisfiedLinkError
- java.lang.VerifyError

Scan the SystemOut log for a class loader exception. When a class loader exception occurs, note the following:

- The accompanying stack trace
- The error message describing the cause of the exception
- The class causing the exception

Example 4-2 on page 174 illustrates an example of a loader exception.

Example 4-2 Example class loader exception

```
[12/04/07 11:05:13:468 EDT] 00000027 ServletWrappe I   SRVE0242I:
[ClassloaderSampleEAR] [/ClassCastExceptionWeb] [LoadClass]:
Initialization successful.
[12/04/07 11:05:13:531 EDT] 00000027 SystemErr     R
java.lang.ClassCastException: bean.Root incompatible with bean.Leaf
```

```
In this example we see that a ClassCastException is thrown by the
Servlet LoadClass within the application ClassLoaderSampleEAR and web
module ClassCastExceptionWeb complaining that the class bean.Root is
incompatible with bean.Leaf.
```

4.2.3 Recreate the problem and collect traces

After you determine that a class loader issue occurred, you need to collect the following trace to begin diagnostics:

- Class loader trace.

  See "Class loader trace" on page 206 for information on enabling this trace.

In addition, if you think you might need to contact IBM support at some point and only want to recreate the problem once, collect the following:

- JVM class loader and bootstrap traces.

  See "JVM class loader and bootstrap traces" on page 207 for information on enabling this trace.

The traces contain exception-specific information that will help you understand the problem.

1. Enable the traces.
2. Stop the application server and clear the existing log files. This ensures that the trace information is fresh.
3. Restart the server.
4. Reproduce the problem, and collect all the log files generated.

Tip: By default the application log files and trace enabled log files are located in the directory *profile_home*\logs*server_name*\.

Now that you have the information you need to diagnose the problem, proceed to the appropriate problem area.

Each class loader problem section will help you do the following:

1. Analyze the class loader trace to find information pertinent to the exception or error you are encountering. This information helps you to identify the specific cause of the class loader exception.
2. Evaluate the data to determine the cause for the exception and error.

3. Examine the possible root causes and solutions for these class loader exceptions and errors.

If you do not find one of these exceptions or need further help, go to “The next step” on page 213 for information on performing online searches using your symptom and collecting data to contact IBM technical support.

4.3 ClassCastException

This section takes you through the process of diagnosing and resolving ClassCastException errors.

4.3.1 Analyze the class loader trace

First, find the ClassCastException in the trace. Note the following:

- The classes that caused the exception
- The line in the application where the error occurs

Next, search the trace for information about the classes referenced in the exception. Note the following:

- The class loader that loaded the class files
- The location of the class files.

Example

In this Example 4-3 , the trace tells you that the ClassCastException is caused by an incompatibility of the class bean.Root with bean.Leaf. The error occurs at line 50 in the application.

Example 4-3 Example ClassCastException

```
[12/04/07 11:05:13:531 EDT] 00000027 SystemErr     R
java.lang.ClassCastException: bean.Root incompatible with bean.Leaf
[12/04/07 11:05:13:531 EDT] 00000027 SystemErr     R   at
servlet.LoadClass.doPost(LoadClass.java:50)
[12/04/07 11:05:13:531 EDT] 00000027 SystemErr     R   at
servlet.LoadClass.doGet(LoadClass.java:36)
[12/04/07 11:05:13:531 EDT] 00000027 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
[12/04/07 11:05:13:531 EDT] 00000027 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
```

```
[12/04/07 11:05:13:531 EDT] 00000027 SystemErr     R   at
com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.j
ava:966)
```

A search for the two classes in the trace, bean.Root and bean.Leaf, provides the following information.

Example 4-4 shows the information for bean.Root.

Example 4-4 bean.Root

```
[12/04/07 11:05:13:437 EDT] 00000027 CompoundClass >  findClass
name=bean.Root this=com.ibm.ws.classloader.CompoundClassLoader@29582958
Entry
[12/04/07 11:05:13:437 EDT] 00000027 ReloadableCla 3   adding path to
reload cache
                                  C:\UtilityJar.jar
[12/04/07 11:05:13:437 EDT] 00000027 CompoundClass 3   class bean.Root
found in SinglePathClassProvider :
com.ibm.ws.classloader.SinglePathClassProvider@14c014c classpath =
C:\UtilityJar.jar
[12/04/07 11:05:13:437 EDT] 00000027 CompoundClass >  loadClass
name=java.lang.Object
this=com.ibm.ws.classloader.CompoundClassLoader@29582958 Entry
[12/04/07 11:05:13:453 EDT] 00000027 CompoundClass 3   loaded bean.Root
from this=
com.ibm.ws.classloader.CompoundClassLoader@29582958
   Local ClassPath:
C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps\jasperch
uiNode02Cell\ClassloaderSampleEAR.ear\ClassCastExceptionWeb.war\WEB-INF
\classes;C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps
\jasperchuiNode02Cell\ClassloaderSampleEAR.ear\ClassCastExceptionWeb.wa
r;C:\UtilityJar.jar
   Delegation Mode: PARENT_FIRST
```

Example 4-5 on page 177 shows the information for bean.Leaf.

Example 4-5 bean.Leaf

```
[12/04/07 11:05:13:453 EDT] 00000027 CompoundClass >  findClass
name=bean.Leaf this=com.ibm.ws.classloader.CompoundClassLoader@29582958
Entry
[12/04/07 11:05:13:453 EDT] 00000027 CompoundClass 3   class bean.Leaf
found in SinglePathClassProvider :
```

```
com.ibm.ws.classloader.SinglePathClassProvider@14c014c classpath =
C:\UtilityJar.jar
[12/04/07 11:05:13:453 EDT] 00000027 CompoundClass <  findClass Exit
[12/04/07 11:05:13:453 EDT] 00000027 CompoundClass 3   loaded bean.Leaf
from this=
com.ibm.ws.classloader.CompoundClassLoader@29582958
   Local ClassPath:
C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps\jasperch
uiNode02Cell\ClassloaderSampleEAR.ear\ClassCastExceptionWeb.war\WEB-INF
\classes;C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps
\jasperchuiNode02Cell\ClassloaderSampleEAR.ear\ClassCastExceptionWeb.wa
r;C:\UtilityJar.jar
   Delegation Mode: PARENT_FIRST
```

4.3.2 Evaluate the trace data

Do the following:

- Determine if the source and target are loaded by the same class loader. If not, see “Class loader for source and target are different” on page 180.
- Using the information from the trace, examine the source code. The source object should be an instance of the target class. If not, see “Source object is not an instance of the target class” on page 179.
- Check the source to see if the failure occurred while performing a narrow operation on EJBs. Example 4-6 shows a sample narrow operation being performed for an EJB remote interface. If a narrow operation is being performed, see “Application improperly performs a narrow operation” on page 180.

Example 4-6 Narrow operation

```
java.lang.Object ejbHome =
initialContext.lookup("java:comp/env/ejb/myEJB");
myEJBHome =
(myEJBHome)javax.rmi.PortableRemoteObject.narrow((org.omg.CORBA.Object)
ejbHome, myEJBHome.class);
```

Example

From the example traces, you know the following:

- The ClassCastException is caused by an incompatibility of the class bean.Root with bean.Leaf.
- The error occurs at line 50 in the application.

- The class loader that loaded the bean.Root class and the bean.Leaf class is the same class loader:

  ```
  SinglePathClassProvider:
  com.ibm.ws.classloader.SinglePathClassProvider@14c014c
  ```

> **Tip:** If the class is located by the Java class loader: extensions class loader, application class loader or the bootstrap class loader, specifying `-Dibm.cl.verbose=<name>` in the generic JVM argument allows you to trace the way these class loaders find and load a given class.
>
> Use the full name of the class, including the package name, for example: `-Dibm.cl.verbose=bean.Root`.
>
> See "Additional class loader diagnostics" on page 211 for more information.

- The files are located in C:\UtilityJar.jar.

  You can also reach this conclusion if you search for the classes with the Class Loader Viewer.

The exception fails on line 50 of the servlet class LoadClass. The source for the sample application has the code shown in Example 4-7 at that location:

Example 4-7 Source code from ClassCastExceptionWeb sample.

```
49      Root r = new Root();
50      Leaf l = (Leaf)r;
```

The code is trying to cast the object `r` to the class bean.Leaf, which is not possible since the object `r` is of the type bean.Root. This problem is addressed in "Source object is not an instance of the target class" on page 179.

4.3.3 ClassCastException root causes

This section lists the possible root causes and possible solutions for a ClassCastException error.

Source object is not an instance of the target class

Ensure that the source class exists within the ancestry of the target class or is the same class. You can determine this by examining the output of the following:

```
System.out.println( source.getClass().getName() + ":" +
target.getClass().getName());
```

Where *source* represents the object being cast and the *target* represents the class being cast to. From the sample application, Example 4-7 on page 179, r is the source object and Leaf is the target class.

Or use a **javap** command as shown in Example 4-8 .

Example 4-8 Using the javap command to investigate ancestry of a class.

```
javap java.util.HashMap
Compiled from “HashMap.java”
public class java.util.HashMap extends java.util.AbstractMap implements
java.util.Map, java.lang.Cloneable, java.io.Serializable {
```

Resolve the problem

After you confirm that this is the cause of the problem, review your code to ensure that it is performing the correct and proper casting for the target class. In the sample, the Leaf class has the ancestry:

```
public class Leaf extends Root
```

It would be more appropriate to have the following:

```
Root r = new Root();
Leaf l = new Leaf();
Root c = (Root) l;
```

Class loader for source and target are different

When the class loader that loaded the source object is different than the class loader that loaded the target class you will have a problem.

Resolve the problem

To resolve this, ensure that the classes are loaded by the same class loader. You should already have this information from reviewing the trace. If they are not, refer to “Using custom JAR files for your application” on page 200 for suggestions on using custom jars in your application and to ensure that the two classes are loaded by the same class loader, for example, by placing the files in the same JAR file if possible.

Application improperly performs a narrow operation

When an application fails to perform, or improperly performs, a narrow operation problem will occur.

The narrow operation must be performed on objects returned from a look up for a remote interface of an EJB (Not for the local interface of an EJB).

Resolve the problem

Make sure that the target class submitted to the narrow method is the exact remote interface of the EJB.

Where to go from here: If these suggestions do not resolve your problem, go to "The next step" on page 213 for information on performing online searches of known class loader issues. The referenced section also contains information on what you need to collect in order to engage IBM support.

4.4 ClassNotFoundException

This section takes you through the process of diagnosing and resolving ClassNotFoundExceptions errors.

4.4.1 Analyze the class loader trace

First, search for the ClassNotFoundException and note the following:

- The class that could not be found
- The line number in the application where the failure occurs

Next, search the trace for information about the class.

- Note the attempts to find the class by each class loader.

Example

In Example 4-9 the ClassNotFoundException is being thrown when trying to locate the class `sample.ReferenceClass`. The failure occurs at line 34 in the application.

Example 4-9 Example ClassNotFoundException

```
[12/04/07 10:56:26:093 EDT] 00000024 SystemErr     R
java.lang.ClassNotFoundException: sample.ReferenceClass
[12/04/07 10:56:26:093 EDT] 00000024 SystemErr     R   at
java.lang.Class.forNameImpl(Native Method)
[12/04/07 10:56:26:093 EDT] 00000024 SystemErr     R   at
java.lang.Class.forName(Class.java:131)
[12/04/07 10:56:26:093 EDT] 00000024 SystemErr     R   at
servlet.LoadClass.doPost(LoadClass.java:34)
```

```
[12/04/07 10:56:26:093 EDT] 00000024 SystemErr     R   at
servlet.LoadClass.doGet(LoadClass.java:26)
[12/04/07 10:56:26:093 EDT] 00000024 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
[12/04/07 10:56:26:093 EDT] 00000024 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
```

A search for sample.ReferenceClass finds the trace entries shown in Example 4-10 . You can see from the trace that the class loaders tried to find the class, but it could not be found.

Example 4-10   sample.ReferenceClass in the class loader trace files

```
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  loadClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@1b501b50 Entry
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  loadClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@5eda5eda Entry
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  findClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@5eda5eda Entry
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  findClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@1b501b50 Entry
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  loadClass
name=java.lang.Throwable
this=com.ibm.ws.classloader.CompoundClassLoader@1b501b50 Entry
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  loadClass
name=java.lang.Throwable
this=com.ibm.ws.classloader.CompoundClassLoader@5eda5eda Entry
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
java.lang.Throwable from parent
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
java.lang.Throwable using classloader=null
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass <  loadClass Exit
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
java.lang.Throwable from parent
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
java.lang.Throwable using classloader=null
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass <  loadClass Exit
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  loadClass
name=javax.servlet.http.HttpServletRequest
this=com.ibm.ws.classloader.CompoundClassLoader@1b501b50 Entry
```

```
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass >  loadClass
name=javax.servlet.http.HttpServletRequest
this=com.ibm.ws.classloader.CompoundClassLoader@5eda5eda Entry
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
javax.servlet.http.HttpServletRequest from parent
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
javax.servlet.http.HttpServletRequest using
classloader=sun.misc.Launcher$AppClassLoader@60e060e0
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass <  loadClass Exit
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
javax.servlet.http.HttpServletRequest from parent
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass 3   loaded
javax.servlet.http.HttpServletRequest using
classloader=sun.misc.Launcher$AppClassLoader@60e060e0
[12/04/07 10:56:26:093 EDT] 00000024 CompoundClass <  loadClass Exit
```

4.4.2 Evaluate the trace data

Using the trace data, you can determine the exception, the class throwing the exception, and the line number where the exception occurred.

- If your application makes use of the class loader API to load the missing class, then do the following:
  - Check to make sure you are using the class loader API correctly. See "Application incorrectly uses a class loader API" on page 184.
  - Make sure that the missing class is visible to the class loader that loaded the parent class. A dependent class of the referenced class must be visible to the same class loader. See "A dependent class is not visible" on page 184.
- If the referenced class is an IBM WebSphere class, see "Access to internal server classes" on page 185.
- If none of the class loaders could find the class, see "Class not available on the classpath" on page 184.

Example

Using the example trace data in Example 4-9 on page 181, you know the following:

- The exception occurred in the LoadClass on line 34. The source for the application contains the following code at that location:

```
34       Class.forName("sample.ReferenceClass");
```

From looking at this line of code, you know the exception occurred when the LoadClass servlet invoked the class loader API to load the sample.ReferenceClass class.

- The class loaders tried to find the class but it could not be found.

In this example, the problem occurred because the class was not on the class path. The problem was resolved by placing the class in a shared library and associating that shared library with the Web module.

4.4.3 ClassNotFoundException root causes

This section lists the possible root causes and possible solutions for ClassNotFoundException errors.

Class not available on the classpath

Ensure that the class is available in the classpath. If the referenced class is in a shared library, ensure that the shared library containing the referenced class is available and associated with the application.

Refer to “Using custom JAR files for your application” on page 200 for suggestions on using custom JAR files in your application.

Application incorrectly uses a class loader API

It is possible to load classes programmatically by obtaining an instance of the class loader and directly loading a class using the loadClass method or by calling Class.forName(class_name, initialize, class_loader) with that class loader. Possible issues for this may be that the name of the class is incorrect, the class is not visible on the classpath of that class loader, or the wrong class loader was engaged.

To correct this, search for the class with the Class Loader Viewer. If it is available, make sure that the correct context is used or the correct API call is used to load the class.

A dependent class is not visible

Dependent classes of the referenced class must be loaded by the same class loader. Search for the class calling the Class.forName method and use the Class Loader Viewer to determine the class loader used to load this class. Search for the dependent classes in the Class Loader Viewer and ensure they are loaded by the same class loader as the referenced class. Refer to “Using custom JAR files for your application” on page 200 for suggestions on using custom jars in your application.

Access to internal server classes

WebSphere Application Server V6.1 gives you the ability to restrict access to internal WebSphere classes so that your applications do not make unsupported calls to WebSphere classes not published in the official WebSphere Application Server API.

This can be configured with the per-server (JVM) setting called `Access to internal server classes`, found by navigating to **Application Server → Server1** in the administrative console.

If this setting is set to `Restrict` and your application makes calls to the WebSphere internal API you can encounter a ClassNotFoundException. Change the setting to `Allow` to correct this. It is, however, recommended that you restrict access to the internal WebSphere API because in the future this restriction may become permanent.

Where to go from here: If these suggestions do not resolve your problem, go to "The next step" on page 213 for information about performing online searches of known class loader issues. This section also contains information on what you will need to collect in order to engage IBM support.

4.5 NoClassDefFoundError

This section takes you through the process of diagnosing and resolving NoClassDefFoundError problems.

4.5.1 Analyze the trace

First, find the NoClassDefFoundError in the trace and note the following:

- The name of the missing referenced class
- The line in the application where the error occurred

Next, search for entries relevant to the class in the trace and note what classloaders were used to search for this class.

Example

In the trace in Example 4-11 you can see the missing referenced class is sample.ReferenceClass. The error occurred on line 7 in the application.

Example 4-11 Example NoClassDefFoundError

```
[12/04/07 10:49:19:687 EDT] 00000024 SystemErr     R
java.lang.NoClassDefFoundError: sample.ReferenceClass
[12/04/07 10:49:19:687 EDT] 00000024 SystemErr     R   at
servlet.InvokeAction.performAction(InvokeAction.java:7)
[12/04/07 10:49:19:687 EDT] 00000024 SystemErr     R   at
servlet.LoadClass.doPost(LoadClass.java:45)
[12/04/07 10:49:19:687 EDT] 00000024 SystemErr     R   at
servlet.LoadClass.doGet(LoadClass.java:33)
[12/04/07 10:49:19:687 EDT] 00000024 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
[12/04/07 10:49:19:687 EDT] 00000024 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
```

The entries for sample.ReferenceClass are shown in Example 4-12 . You can see that the class loaders tried to find the class, but it could not be found.

Example 4-12 Trace entries for sample.ReferenceClass

```
[12/04/07 10:49:19:687 EDT] 00000024 CompoundClass >  loadClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@43484348 Entry
[12/04/07 10:49:19:687 EDT] 00000024 CompoundClass >  loadClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@60026002 Entry
[12/04/07 10:49:19:687 EDT] 00000024 CompoundClass >  findClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@60026002 Entry
[12/04/07 10:49:19:687 EDT] 00000024 CompoundClass >  findClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@43484348 Entry
[12/04/07 10:49:19:687 EDT] 00000024 SystemErr     R
java.lang.NoClassDefFoundError: sample.ReferenceClass
```

4.5.2 Evaluate the trace data

Search for the missing class file on the file system. If you can find it, note the location and the name of the JAR file the class resides in.

Use the Class Loader Viewer (table mode) to determine if the JAR file or the location exists on the class path of the class loaders in WebSphere.

- If it is on the class path, check the class to ensure it can be loaded. See “Class cannot load” on page 187 for possible solutions.

- If it is not on the class path, ensure that the missing class is on the class path and available to the class loader to load. See “Class not available on the classpath” on page 187.

Example

From the example trace, you know the following:

- The NoClassdefFoundError occurs because the class loader cannot locate the class sample.ReferenceClass.
- The error occurs at line 7 of the InvokeAction.

In the source code for the InvokeAction class, you can see line 7 creates an instance of the class ReferenceClass.

```
7      ReferenceClass root = new ReferenceClass(“Value returned");
```

However the class loader in WebSphere cannot locate the ReferenceClass, so it throws the NoClassDefFoundException.

In this example, the problem occurred because the class was not on the class path. The problem was resolved by placing the class in a shared library and associating that shared library with the Web module.

4.5.3 NoClassDefFoundError root causes

This section lists the possible root causes and possible solutions for a NoClassDefFoundError conditions.

Class not available on the classpath

Ensure that the library is available in the classpath. If the referenced class is in a shared library, check that the shared library containing the referenced class is available and associated with the application.

Refer to “Using custom JAR files for your application” on page 200 for suggestions on using custom jars in your application.

Class cannot load

For various reasons the class could not be loaded. Possible reasons include the following:

- Failure to load the dependent class
- The dependent class has a bad format
- The version number of a class

Ensure these are correct.

See “Using custom JAR files for your application” on page 200 for more information.

Where to go from here: If these suggestions do not resolve your problem, go to “The next step” on page 213 for information about performing online searches of known class loader issues. This section also contains information on what you will need to collect in order to engage IBM support.

4.6 NoSuchMethodError and IllegalArgumentException

This section takes you through the process of diagnosing and resolving NoSuchMethodErrors and IllegalArgumentException errors.

4.6.1 Analyze the trace

A NoSuchMethodError looks like Example 4-13 on page 188.

Find the error, and note the following:

- The class the exception is referencing
- The missing method and parameter
- The line in the application where the error occurred

Search for the class in the trace files, and note the following:

- The class loader loading this class
- The location of the class file

Example

In Example 4-13 on page 188 the class `sample.ReferenceClass` is missing the method `getText` with parameter `java.lang.String`. The error occurs on line 8 in the application.

Example 4-13 Example NoSuchMethodError

```
[12/04/07 11:00:42:437 EDT] 00000024 SystemErr     R
java.lang.NoSuchMethodError:
sample/ReferenceClass.getText()Ljava/lang/String;
[12/04/07 11:00:42:437 EDT] 00000024 SystemErr     R   at
servlet.InvokeAction.performAction(InvokeAction.java:8)
[12/04/07 11:00:42:437 EDT] 00000024 SystemErr     R   at
servlet.LoadClass.doPost(LoadClass.java:45)
```

```
[12/04/07 11:00:42:437 EDT] 00000024 SystemErr     R   at
servlet.LoadClass.doGet(LoadClass.java:33)
[12/04/07 11:00:42:437 EDT] 00000024 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
[12/04/07 11:00:42:437 EDT] 00000024 SystemErr     R   at
javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
[12/04/07 11:00:42:437 EDT] 00000024 SystemErr     R   at
com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.j
ava:966)
```

An IllegalArgumentException has a similar format. It will reference a class and a particular message related to the exception. An IllegalArgumentException may complain of the incorrect argument being passed to a method.

In Example 4-14 you can see that sample.ReferenceClass is loaded by the class loader `SinglePathClassProvider : com.ibm.ws.Class loader.SinglePathClassProvider@15761576` and is located in the `C:\MissingMethodUtilityJar.jar` file.

Example 4-14 Trace entries showing the class being loaded

```
[12/04/07 11:00:42:421 EDT] 00000024 CompoundClass >  loadClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@22462246 Entry
[12/04/07 11:00:42:421 EDT] 00000024 CompoundClass >  loadClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@48c248c2 Entry
[12/04/07 11:00:42:421 EDT] 00000024 CompoundClass >  findClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@48c248c2 Entry
[12/04/07 11:00:42:421 EDT] 00000024 CompoundClass >  findClass
name=sample.ReferenceClass
this=com.ibm.ws.classloader.CompoundClassLoader@22462246 Entry
[12/04/07 11:00:42:421 EDT] 00000024 ReloadableCla 3   adding path to
reload cache
                                 c:\MissingMethodUtilityJar.jar
[12/04/07 11:00:42:437 EDT] 00000024 CompoundClass 3   class
sample.ReferenceClass found in SinglePathClassProvider :
com.ibm.ws.classloader.SinglePathClassProvider@15761576 classpath =
c:\MissingMethodUtilityJar.jar
[12/04/07 11:00:42:437 EDT] 00000024 CompoundClass <  findClass Exit
[12/04/07 11:00:42:437 EDT] 00000024 CompoundClass 3   loaded
sample.ReferenceClass from this=
com.ibm.ws.classloader.CompoundClassLoader@22462246
```

```
   Local ClassPath:
C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps\jasperch
uiNode02Cell\ClassloaderSampleEAR.ear\NoSuchMethodErrorWeb.war\WEB-INF\
classes;C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps\
jasperchuiNode02Cell\ClassloaderSampleEAR.ear\NoSuchMethodErrorWeb.war;
C:\MissingMethodUtilityJar.jar
   Delegation Mode: PARENT_FIRST
[12/04/07 11:00:42:437 EDT] 00000024 CompoundClass 3   loaded
sample.ReferenceClass using classloader=
com.ibm.ws.classloader.CompoundClassLoader@22462246
   Local ClassPath:
C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps\jasperch
uiNode02Cell\ClassloaderSampleEAR.ear\NoSuchMethodErrorWeb.war\WEB-INF\
classes;C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\installedApps\
jasperchuiNode02Cell\ClassloaderSampleEAR.ear\NoSuchMethodErrorWeb.war;
C:\MissingMethodUtilityJar.jar
   Delegation Mode: PARENT_FIRST
[12/04/07 11:00:42:437 EDT] 00000024 CompoundClass <  loadClass Exit
```

4.6.2 Evaluate the trace data

Using the information gathered in the trace, determine if the class loaded by the class loader contains the expected version of the class.

Use the Class Loader Viewer to determine the location of the class referenced and to make sure that the class loaded is the correct one.

If the class loaded by the class loader in WebSphere does not contain the method or the argument referenced by the source code, see "Different version of classes" on page 191.

Example

From the trace entries in the example, you know the following:

- The class sample.ReferenceClass is missing the getText method with parameter java.lang.String.
- The class sample.ReferenceClass is located in the following file, C:\MissingMethodUtilityJar.jar.
- The method is being called from the InvokeAction class on line 8 and is invoking the method sample/ReferenceClass.getText(String text).

This tells you that `MissingMethodUtilityJar.jar` contains the ReferenceClass class file that does not have the method getText(String). The problem was resolved by placing the correct version of the class with the correct method in a shared library and associating that shared library with the Web module.

4.6.3 NoSuchMethodError, IllegalArgumentException root causes

This section lists the possible root causes and possible solutions for a NoSuchMethodError or IllegalArgumentException conditions.

Different version of classes

An incorrect version of the class is being referenced.

It may be possible that there are multiple versions of the class or JAR files available on the class path of the class loaders in WebSphere. If they are not needed for other applications deployed to the server you can remove the incorrect versions of the class. If they are needed, use the class loader delegation mode to ensure the correct version is loaded. See “Class loading/delegation mode” on page 170 for more information.

If the class is in a shared library, ensure that the correct files are associated with the shared library and with the application.

Refer to “Using custom JAR files for your application” on page 200 for suggestions on using custom jars in your application.

Example scenario

For example if there are two copies of the referenced class file, one loaded by the WebSphere extensions classloader and one available to the application classloader, ensure that the class loader delegation mode of `Parent Last` is set to pick up the class within your application instead of the one on the server.

Where to go from here: If these suggestions do not resolve your problem, go to “The next step” on page 213 for information about performing online searches of known class loader issues. This section also contains information on what you will need to collect in order to engage IBM support.

4.7 UnsatisfiedLinkError

This section takes you through the process of diagnosing and resolving UnsatisfiedLinkError problems.

4.7.1 Analyze the trace

An UnsatisfiedLinkError message will resemble Example 4-15 . The error message of the UnsatisfiedLinkError will have the name of the missing native library.

- Note the native library referenced.

Example 4-15 Example UnsatisfiedLinkError message.

```
java.lang.UnsatisfiedLinkError mynativelibrary.so
```

In the example, the library is `mynativelibrary`.

4.7.2 Evaluate the trace data

Other than the library name, the trace does not provide much information. There are several possibilities for the root cause of the problem, but you need to check each possibility until you find your problem.

The possible root causes for this problem are as follows:

- The native library is not referenced properly. See “Configuration error” on page 193.
- The library is not visible to the application. See “Library is not visible” on page 193 for suggestions on how to make it visible.
- The native library is loaded more than once. See “Native library is already loaded” on page 193.
- The dependent native libraries for the referenced native library are not successfully loaded. See “Dependent native library configuration” on page 195.
- There are problems with the JVM itself in loading the native libraries. The method System.mapLibraryName is used to return the proper extension for referencing a native library file in the various operating systems.

  See “System.mapLibraryName returns the wrong library file” on page 196 for possible actions to resolve this.

4.7.3 UnsatisfiedLinkError root causes

This section lists the possible root causes and possible solutions for an UnsatisfiedLinkError condition.

Configuration error

A configuration error is usually one of the following:

- Incorrect library extension:
  - Windows: A library has the dynamic link library name library_name.dll.
  - AIX, HP-UX, Solaris, Linux: A library has the name library_name.so or library_name.a.
- System.loadLibrary is passed an incorrect parameter:
  - Windows: To load Name.dll, Name is passed to the loadLibrary method.
  - AIX, HP-UX, Solaris, Linux: To load libName.so or libName.a, libName is passed to the loadLibrary method.

Correct the native library reference in your application or the shared library reference to resolve this.

Library is not visible

Ensure that the native library is available on the Java library path (java.library.path). During startup of the WebSphere Application Server, the SystemOut log prints the Java library path (java.library.path) variable and lists the directories and libraries that it will load. Check this to make sure your native library is there.

You can also configure a shared library for the native library and associate it with the application. Specify the path to the native library in the Native Library Path field and the class or JAR file referencing the native library in the Classpath field.

Refer to “Using custom JAR files for your application” on page 200 for suggestions on using custom JAR files in your application.

Native library is already loaded

If the native library was already loaded by an application and the same application tries to load it again, this can cause this error. One possible scenario for this to occur is if the application is restarted and it tries to load the native library again. WebSphere has no control over the native code and if it does not unload and load properly the application may fail.

To determine if the application was restarted. Check the SystemOut log for the following sequence of logs. See Example 4-16 for sample log output.

Example 4-16 Restarting an application

```
[03/05/07 13:47:50:781 EDT] 0000002f ApplicationMg A   WSVR0220I: 
Application stopped: ClassloaderSampleEAR
```

```
[03/05/07 13:47:57:031 EDT] 00000030 ApplicationMg A   WSVR0200I:
Starting application: ClassloaderSampleEAR
[03/05/07 13:47:57:046 EDT] 00000030 ApplicationMg A   WSVR0204I:
Application: ClassloaderSampleEAR  Application build level: Unknown
[03/05/07 13:47:58:078 EDT] 00000030 WebGroup      A   SRVE0169I:
Loading Web Module: ClassNotFoundExceptionWeb.
[03/05/07 13:47:58:140 EDT] 00000030 VirtualHost   I   SRVE0250I: Web
Module ClassNotFoundExceptionWeb has been bound to
default_host[*:9081,*:80,*:9444,*:5063,*:5062,*:443].
[03/05/07 13:47:58:218 EDT] 00000030 WebGroup      A   SRVE0169I:
Loading Web Module: ClassCastExceptionWeb.
[03/05/07 13:47:58:343 EDT] 00000030 VirtualHost   I   SRVE0250I: Web
Module ClassCastExceptionWeb has been bound to
default_host[*:9081,*:80,*:9444,*:5063,*:5062,*:443].
[03/05/07 13:47:58:390 EDT] 00000030 WebGroup      A   SRVE0169I:
Loading Web Module: NoSuchMethodErrorWeb.
[03/05/07 13:47:58:468 EDT] 00000030 VirtualHost   I   SRVE0250I: Web
Module NoSuchMethodErrorWeb has been bound to
default_host[*:9081,*:80,*:9444,*:5063,*:5062,*:443].
[03/05/07 13:47:58:546 EDT] 00000030 WebGroup      A   SRVE0169I:
Loading Web Module: NoClassDefFoundErrorWeb.
[03/05/07 13:47:58:640 EDT] 00000030 VirtualHost   I   SRVE0250I: Web
Module NoClassDefFoundErrorWeb has been bound to
default_host[*:9081,*:80,*:9444,*:5063,*:5062,*:443].
[03/05/07 13:47:58:687 EDT] 00000030 ApplicationMg A   WSVR0221I:
Application started: ClassloaderSampleEAR
```

To resolve this, ensure that the library is only loaded once. To achieve this you can package a class containing a static call to this native library in a utility JAR file and package it in a shared library to be associated with a server class loader. This ensures that the native library is loaded only once for each Java virtual machine regardless of the application life cycle.

Example 4-17   Sample native library loader

```
Sample Class:
public class LibLoader {
   static {System.loadLibrary(MyNativeLib);}
   public LibLoader();
}
```

After you create a native library loader, you can create and configure a shared library for this class and your native library. Specify the JAR file containing the native library loader in the Classpath field, and specify the path to your native library in the Native Library Path field.

Now create a class loader with the server and associate the shared library to this class loader.

1. In the administrative console, navigate to **Servers** → **Application servers** → ***server_name*** → **Server Infrastructure/Java and Process Management** select **Class loader**.
2. Click **New**, and select the class loader order from the drop-down according to the needs of your application: `Classes loaded with parent class loader first` or `Classes loaded with application class loader first`.
3. Refer to “Class loading/delegation mode” on page 170 for more information about these options.
4. Under Additional Properties, select **Shared library references**, and create a new reference.
5. Select the shared library from the drop down, and click **OK**.
6. Save the server configuration.

Modify your application code to use the native library loader to load your native libraries.

Dependent native library configuration

Dependent native libraries of the native library must be loaded by the JVM class loader. Therefore any dependent native libraries must be on the Java library path (LIBPATH). This is because when the JVM loads the native library it can only call the JVM class loader to resolve the dependency and the JVM class loader can only reference the Java library path to find referenced dependent native libraries.

To determine if your native library is on the Java library path, you can review the SystemOut log from the WebSphere Application Server. During startup it prints the Java library path (java.library.path) and lists the directories and libraries that it will load.

If it is not available, you can append the path to the native library to the platform-specific native library environment variable or to the java.library.path system property of the server process definition.

The native library environment variable is as follows:

- Windows: PATH
- Linux: LD_LIBRARY_PATH
- AIX: LIBPATH
- HP-UX: SHLIB_PATH
- Solaris: LD_LIBRARY_PATH

Use the following steps to set the java.library.path JVM system property:

1. Navigate to **Server** → **Application server** → ***server_name*** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine** → **Custom Properties**.
2. Create a new custom property.
3. Specify the java.library.path as the name and the path to your native library as the value.
4. Save your server configuration.

System.mapLibraryName returns the wrong library file

When loading a shared library, the JVM calls mapLibraryName(libName) to convert libName to a platform specific name. This method may return a file name with an improper extension. For example, in a UNIX environment it may return libName.so rather than libName.a.

Write a program to call System.mapLibraryName() to verify that it returns the correct value. If it does not return the correct value engage IBM Support.

Where to go from here: If these suggestions do not resolve your problem, go to “The next step” on page 213 for information on performing online searches of known class loader issues. This section also contains information on what you will need to collect in order to engage IBM support.

4.8 VerifyError

This section takes you through the process of diagnosing and resolving VerifyError problems.

4.8.1 Analyze the trace

Search for the VerifyError in the trace and note the following:

- The class
- The method
- The error message

Example

A VerifyError will resemble Example 4-18 . In this example, the class is `MyClass`, the method is `MyMethod` and the error message is `incompatible object argument for method call`.

Example 4-18 Example VerifyError.

```
java.lang.VerifyError: (class: MyClass, method: MyMethod signature:
(I)V) incompatible object argument for method call
```

Collect Class Loader Viewer information

Use the Class Loader Viewer to determine how the class is loaded.

4.8.2 VerifyError root causes

VerifyError occurs when the byte code verification process, during class loading, fails due to internal inconsistency or security problems. The error message associated with the VerifyError will help you identify the class and the cause of the error.

Different version of class

A VerifyError is usually the result of loading an incorrect version of the class or library. The information collected from the Class Loader Viewer can help determine the location of the suspect class or library referenced.

If the class is in a shared library, ensure that the correct files are associated with the shared library and with the application.

Refer to “Using custom JAR files for your application” on page 200 for suggestions on using custom jars in your application.

Example scenario

For example if there are two copies of the referenced class file, one loaded by the WebSphere extensions classloader and one available to the application classloader, ensure that the class loader delegation mode of **Class loaded with application class loader first** picks up the class within your application instead of the one on the server.

Where to go from here: If these suggestions do not help, collect the class loader mustgather and engage IBM support. Refer to “Contact IBM” on page 214 for instructions.

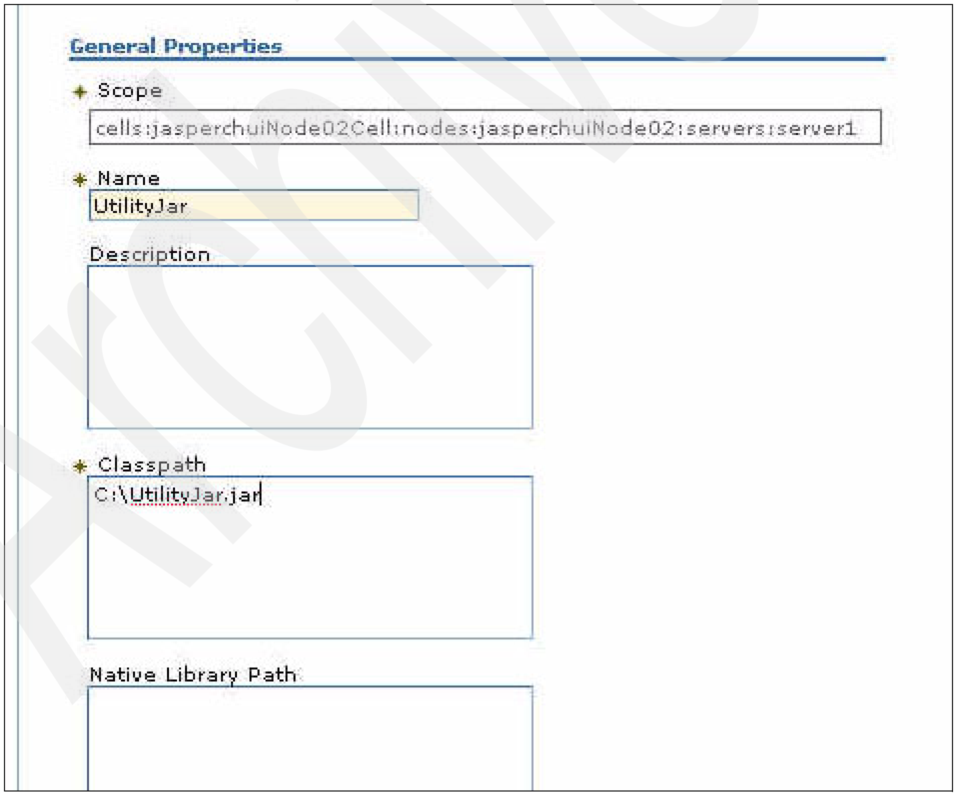
4.9 Configuring shared libraries

Following are instructions to configure a shared library and to associate it with an application.

4.9.1 Create the shared library

To create a shared library, do the following:

1. In the administrative console, navigate to **Environment** → **Shared Libraries**.
2. Select the scope of the shared library. You have the option to choose All, Cell, Node, and Node/Server. This allows the shared library to be visible to the applications and servers residing in these scopes.
3. Click **New**, and specify the name of the shared library, the classpath, or the native classpath to the location of the utility JAR file or utility library. See Figure 4-6 on page 198 for an example.


General Properties

* Scope
cells:jasperchuiNode02Cell:nodes:jasperchuiNode02:servers:server1

* Name
UtilityJar

Description

* Classpath
C:\UtilityJar.jar

Native Library Path

Figure 4-6 Configuring shared library options

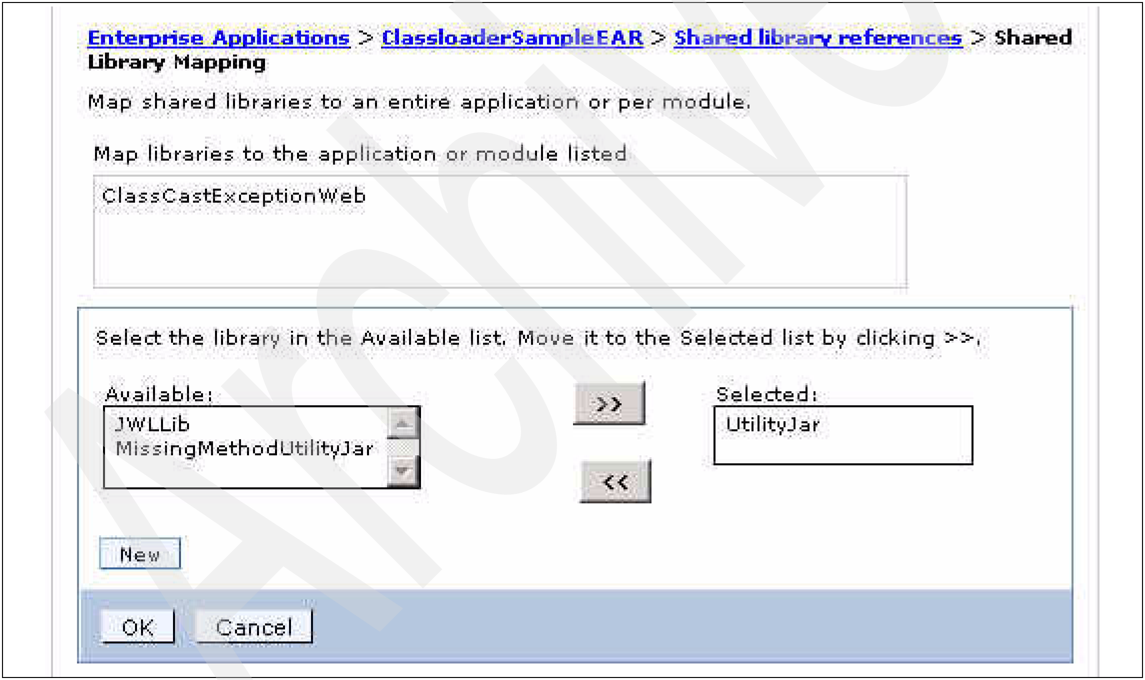
Associate the shared library with an application

To associate the shared library with an application, do the following:

1. In the administrative console, navigate to **Enterprise Application** → **Enterprise Applications**.
2. Click the application name to open the configuration view.
3. Select **Shared library references** under the References section.

   You will see the option to configure a shared library and associate it with a particular application or a Web module.
4. Enable the check box located beside the application or the Web module, and click **Reference shared libraries**.
5. Select the shared library you created, and click the arrow to associate it with this Web module.

   See Figure 4-7 on page 199 for an example.



Enterprise Applications > ClassloaderSampleEAR > Shared library references > **Shared Library Mapping**

Map shared libraries to an entire application or per module.

Map libraries to the application or module listed

ClassCastExceptionWeb

Select the library in the Available list. Move it to the Selected list by clicking >>.

Available:
JWLLib
MissingMethodUtilityJar

Selected:
UtilityJar

Figure 4-7 Associating shared library with a Web module of an application

6. Click **OK**.
7. Click **OK** again to complete the action.
8. Save the configuration.

4.10 Using custom JAR files for your application

Class loader exceptions often occur due to the incorrect location of a common utility class or utility JAR file or the configuration of the server that uses it. Because every application is unique and its requirements are different, it is not simple to determine where best to place a utility file.

This section discusses some possible scenarios. It will help you determine the best place for your utility file and how to configure the WebSphere Application Server to use it.

First, you should have an understanding of what is considered a utility file and where *not* to place these files.

Requirements for a utility file

The basic requirement for a class or JAR file to be a utility file is that *it cannot depend on any application that may be using it*. With this caveat, it is easier to maintain and to allow modifications to the utility files without having to modify the application itself. In addition this condition allows for utility files to be overridden and allows for multiple versions of the utility file to exist in your environment. A utility file can depend on other utility files, but this causes additional complications to the administration of your environment.

Where you should not place utility files

In deciding where best to place your utility files, it is important to recognize that these files should *not* be included in the WebSphere Application Server's environment.

For example: *app_server_root*/lib, *app_server_root*/lib,/ext* , *app_server_root*/java (including any subdirectories), or the JVM classpath.

Adding utility files to those directories can cause problems with the WebSphere runtime environment and can cause unexpected results, including the overwriting of WebSphere classes that can be detrimental to the overall functionality of the server.

Class loader policy options

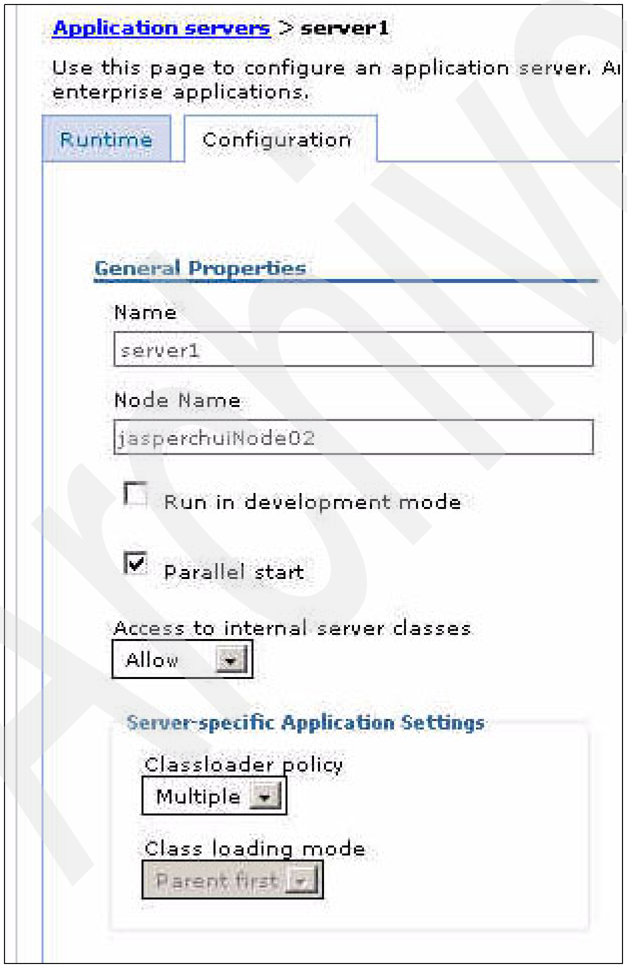
Now that you understand where not to place these files you can look at options for where you can place them. Start by examining the class loader policy options in WebSphere Application Server V6.1.

Server class loader policy settings

To configure the server-specific application settings, navigate to **Application Server** → ***server_name*** (Figure 4-8).

By default the WebSphere server class loader policy is set to `Multiple`, meaning each application will receive its own class loader for loading EJBs, utility JARs, and shared libraries. The alternative is to set the policy to `Single`, meaning there is a single class loader for all the applications on the server.

You also set the class loading mode here, selecting either `Parent first` to load parent classloader files first or `Parent last` to load application classloader files first.



Application servers > server1

Use this page to configure an application server. A
enterprise applications.

General Properties

Name: server1

Node Name: jasperchuiNode02

☐ Run in development mode

✓ Parallel start

Access to internal server classes: Allow

Server-specific Application Settings

Classloader policy: Multiple

Class loading mode: Parent first

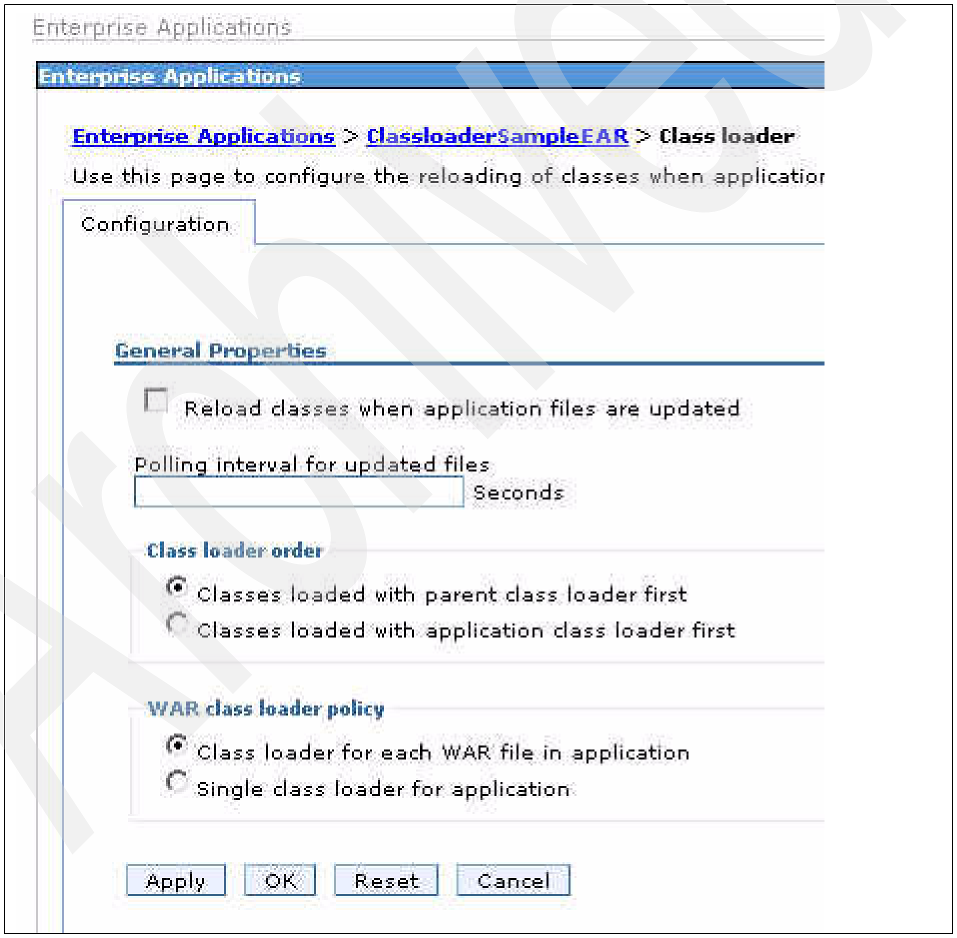
Figure 4-8 Configuring Server-specific Application Setting

Application class loader settings

To configure the application class loader, navigate to **Enterprise Application** → ***application_name*** (Figure 4-9). You can update both the class loader order and the WAR class loader policy.

The default configuration for the application class loader is `Class loaded with parent class loader first`. The alternative option is `Class loaded with application class loader first.` These options are similar to server class loading mode options.

In addition, you have the option to select the WAR class loader policy. You can choose to have a class loader for each WAR file in the application or to have one class loader for the entire application.


Enterprise Applications

Enterprise Applications

Enterprise Applications > ClassloaderSampleEAR > Class loader

Use this page to configure the reloading of classes when applicatior

Configuration

General Properties

☐ Reload classes when application files are updated

Polling interval for updated files
[ ] Seconds

Class loader order

[✓ Classes loaded with parent class loader first]
Classes loaded with application class loader first

WAR class loader policy

[✓ Class loader for each WAR file in application]
Single class loader for application

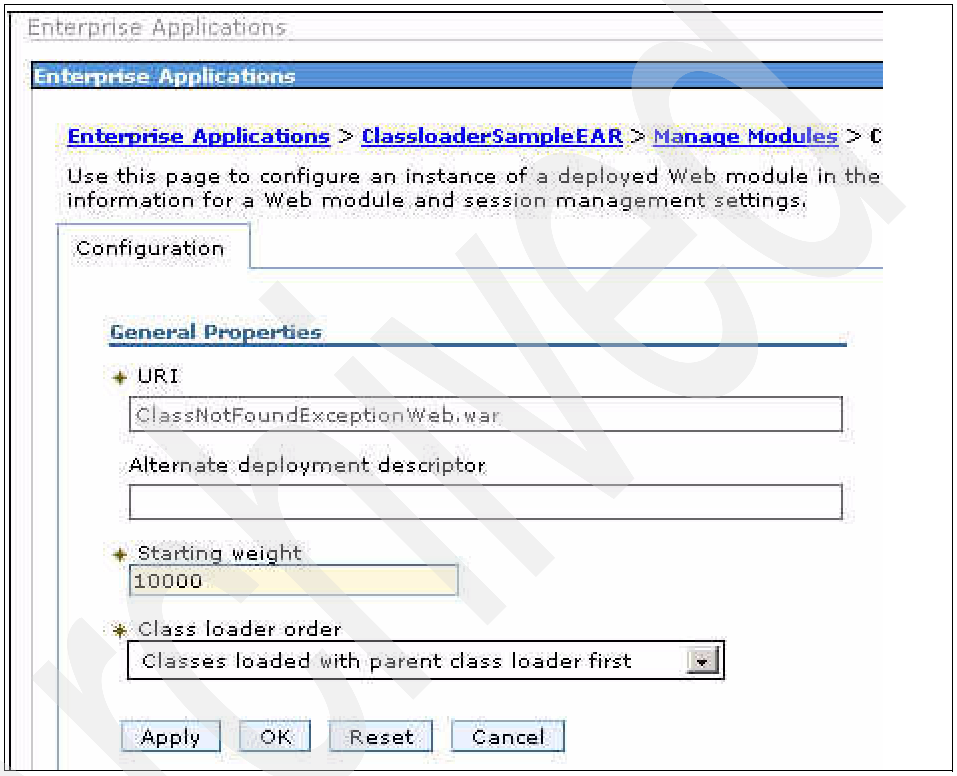
Apply OK Reset Cancel

Figure 4-9 Configuring Application class loader

Web module class loader settings

You can also set the class loader policy at the WAR file. The options are to load classes from the parent class loader first or last.

To configure the WAR class loader, navigate to **Enterprise Application** → ***application_name*** → **Manage Modules** → ***Web_module*** → **Class loader order**. (Figure 4-10).


Enterprise Applications

Enterprise Applications

Enterprise Applications > ClassloaderSampleEAR > Manage Modules > C

Use this page to configure an instance of a deployed Web module in the information for a Web module and session management settings.

Configuration

General Properties

* URI
ClassNotFoundExceptionWeb.war

Alternate deployment descriptor

* Starting weight
10000

* Class loader order
Classes loaded with parent class loader first

Figure 4-10 Configuring Web module class loader

Refer to the following article in the WebSphere Information Center for more information about class loader options:

- *Class loaders*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_classload.html`

Where you should place utility files

Following is a decision tree to help you identify how best to use common application files.

Will you need to deploy different versions of this utility file?

1. **If no**, will this utility file be shared by more than one enterprise application?
   a. **If yes**, will visibility to this utility file cause any other applications installed to the same server to malfunction?
      i. **If yes**, configure a shared library in the server scope, and associate it with the application.
      ii. **If no**, configure a shared library at the server, node, or cell level scope, depending on where the applications using these shared libraries are located, and associate it with all the applications that will use it.

         For example, if you have two applications using the shared library, in a cell on two different servers, create the shared library at the cell level scope so that the shared library is visible to those applications.
   b. **If no**, will multiple modules within the application need this utility file?
      i. **If yes**, place the utility file in the root directory of the EAR, and modify the manifest.mf file to reference the utility JAR.

         You can modify the `manifest.mf` file using the Application Server Toolkit, Rational Application Developer or by modifying the file directly. We recommend that you use an editor.

         To use an editor, import the EAR into the work space and import the utility file to the root of the EAR. From the context of the reference module (Web or EJB), select **Properties** → **J2EE Module Dependencies** → **J2EE Modules** and check the UtilityJar.jar option.

         To modify the manifest.mf manually, open the *reference_module*\META-INF\manifest.mf in a text editor, and add the reference JAR file to the classpath.

         Refer to the following Application Server Toolkit Information Center article for additional information:

         Specify dependent JAR files or modules at the following Web site

         `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.etools.j2ee.doc/topics/tjdepend.html`

         Example 4-19 shows a sample manifest.mf file. In this example, UtilityJar.jar is located in the root directory of the EAR project, and UtilityJar2.jar is located in the folder *mypath* under the root directory of the EAR.

Example 4-19 Example manifest.mf to add UtilityJar.jar to the application classpath.

```
Manifest-Version: 1.0
Class-Path: UtilityJar.jar mypath/UtilityJar2.jar
```

ii. **If no**, place the utility JAR file in the following location:

   EJB module: *EAR_file/EJB_jar/*

   WAR module: *EAR_file/WAR_file/*WEB-INF/lib/

2. **If yes**, configure a shared library for each version of the utility JAR file, and associate them with the application or Web module.

After you place the utility file in the correct location, you can alter the class loader settings to use a different version of a utility file that is already included with WebSphere Application Server. For example, if you want to use a newer version of a library located in your application, you can configure the class loader policy to `Class loaded with application class loader first` so that it picks up the utility files in your application first rather than the one on the server.

Class loader options also provide flexibility in your ability to override existing libraries included with WebSphere.

Tip: When using your own version of Xerces and Xalan, you can also use the `Parent Last` class loader policy to load the version included with your application instead of the one included with the SDK.

Class loaders for custom logging: It is common for users to want to use their own logging mechanism for their applications. Refer to the following documentation in the WebSphere Information Center for more information on configuring the class loaders to use custom logging.

- Jakarta Commons Logging

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/ctrb_classload_jcl.html`

- Configurations for the WebSphere Application Server logger

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/ctrb_classload_jcl_conf.html`

4.11 Collecting diagnostic data

This section discusses how to collect traces relevant to class loader problems. It also includes instructions for using the Class Loader Viewer.

4.11.1 Class loader-specific traces

This section tells you how to collect traces for class loader problems.

Class loader trace

Use the following steps to collect a class loader trace for WebSphere Application server Version 6.1:

1. Log on to the administrative console.
2. In the left navigational panel, expand **Troubleshooting**. Click **Logs and Trace**.
3. Select the application server to be traced, and then on the next page click the **Diagnostic Trace** link.
4. Select the **Configuration tab**.
5. Select the Enable Log property.
6. Under **Trace Output**, select the **File** radio button, and specify the file name. Also Increase the Maximum file size to 100 MB, and increase the Maximum number of historical files to 10.

Note: The name and location of the output file, by default, is ${SERVER_LOG_ROOT}/trace.log. It outputs to *profile_home*\logs\server_name\trace.log.

7. Select **Basic (Compatible) Trace Output Format**.
8. Navigate to **Logging and Tracing** > ***application_server*** > **Change Log Detail Levels**.
9. Under the **Configuration tab**, specify the trace string suggested.

Tip: For class loaders the tracestring is `com.ibm.ws.classloader.*=all`.

10. Click **Apply** and **OK**. Save your configuration. Select the **Synchronize changes with Nodes** option if you are using a distributed environment.

JVM class loader and bootstrap traces

To enable the Java Virtual Machine (JVM) class loader and bootstrap traces for Version 6.1, do the following:

1. From the administrative console, select **Servers > Application servers**, and select the problem server.
2. On the right side, under Server Infrastructure, expand **Java and Process Management**.
3. Click **Process Definition**.
4. Under **Additional Properties**, select **Java Virtual Machine**.
5. Enable the following Properties by selecting the check boxes located on the left of the property.
   - `Verbose class loading`
   - `Verbose JNI`
6. Click **Apply**.
7. On right side under **Additional Properties**, click **Custom Properties**.
8. Click **New**.
9. Enter the following custom properties:
   - Name: `ws.ext.debug`
   - Value: `true`
10. Click **Apply** and **OK**. Save your configuration. Select the **Synchronize changes with Nodes** option if you are using a distributed environment.

4.11.2 Using the Class Loader Viewer

The Class Loader Viewer in the administrative console can be used to determine how a class is loaded by the class loaders in the WebSphere Application Server.

Enable the Class Loader Viewer service:

If the Class Loader Viewer service is not enabled, the Class Loader Viewer only displays the hierarchy of class loaders and their classpaths, but not the classes actually loaded by each of the class loaders. This also means that the search capability of the Class Loader Viewer is lost.

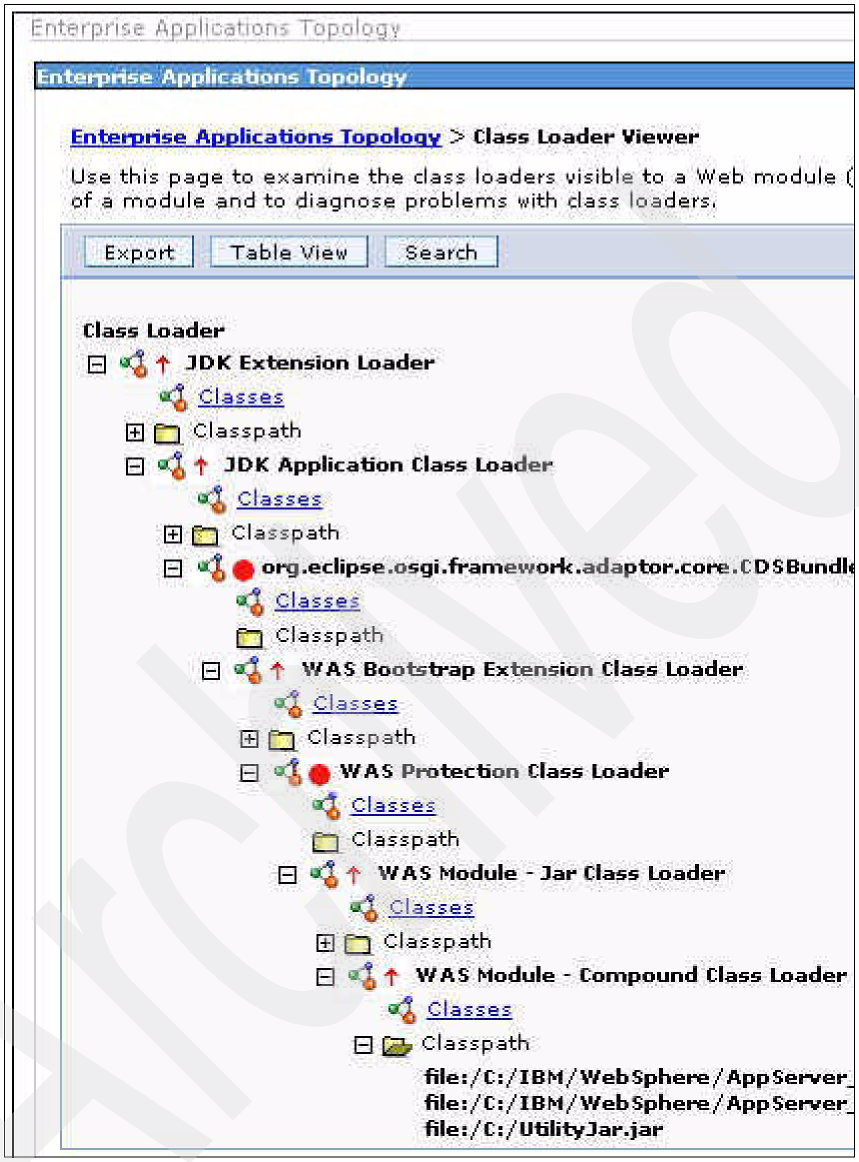
To enable the Class Loader Viewer Service, select **Servers** → **Application Servers** → ***server_name***, and then click the **Class Loader Viewer Service** under the **Additional Properties** link. Next, select **Enable service at server startup**. Restart the application server for the setting to take effect.

From the administrative console, navigate to **Troubleshooting** → **Class Loader Viewe**r, and expand the <server_name> tree until you see the application and its modules.

With the Class Loader Viewer you have two options to view the classes associated with the Web module, the default tree view (Figure 4-11 on page 209) or the Table View (Figure 4-12 on page 210). In both options the viewer lists the class loader class, the JAR files on each of the class loader's classpath, if the Class Loader Viewer Service is enabled, and the classes loaded by the class loaders.

As an example, search for the classes bean.Leaf and bean.Root used to illustrate the ClassCastException error.

From the administrative console, navigate to **Troubleshooting** → **Class Loader Viewer**, and expand the topology tree until you see the ClassloaderSampleEAR and its associated Web modules. Next, select the ClassCastExceptionWeb.war.


Enterprise Applications Topology

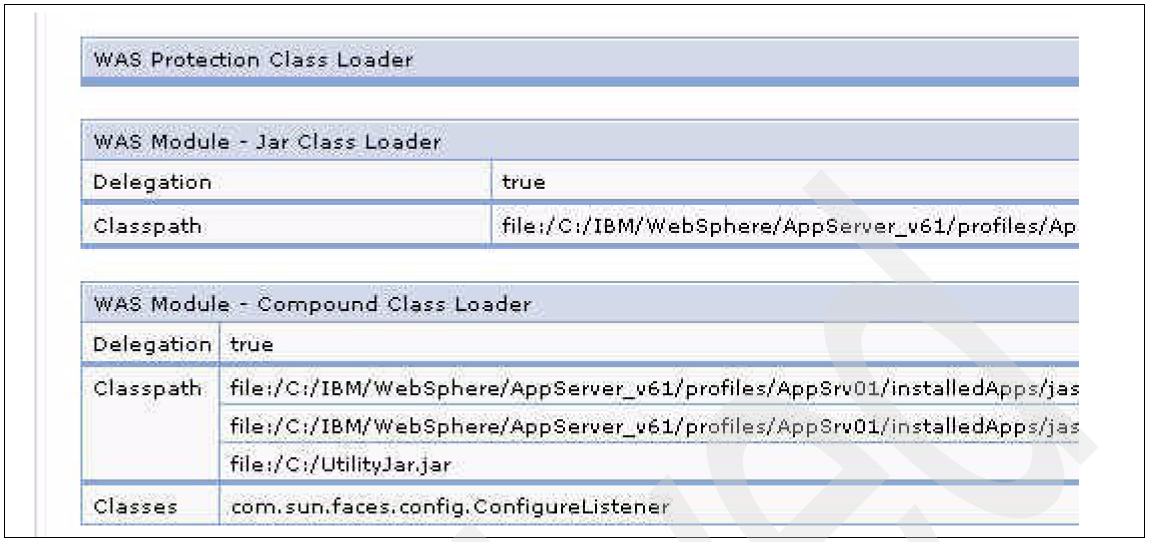
Enterprise Applications Topology

Enterprise Applications Topology > Class Loader Viewer

Use this page to examine the class loaders visible to a Web module (
of a module and to diagnose problems with class loaders.

```
Class Loader
⊟ JDK Extension Loader
    Classes
  ⊞ Classpath
  ⊟ JDK Application Class Loader
      Classes
    ⊞ Classpath
    ⊟ org.eclipse.osgi.framework.adaptor.core.CDSBundl
        Classes
        Classpath
      ⊟ WAS Bootstrap Extension Class Loader
          Classes
        ⊞ Classpath
        ⊟ WAS Protection Class Loader
            Classes
            Classpath
          ⊟ WAS Module - Jar Class Loader
              Classes
            ⊞ Classpath
            ⊟ WAS Module - Compound Class Loader
                Classes
              ⊟ Classpath
                  file:/C:/IBM/WebSphere/AppServer_
                  file:/C:/IBM/WebSphere/AppServer_
                  file:/C:/UtilityJar.jar
```

Figure 4-11 Default Tree mode for the Class Loader Viewer



WAS Protection Class Loader	

WAS Module - Jar Class Loader	
Delegation	true
Classpath	file:/C:/IBM/WebSphere/AppServer_v61/profiles/Ap

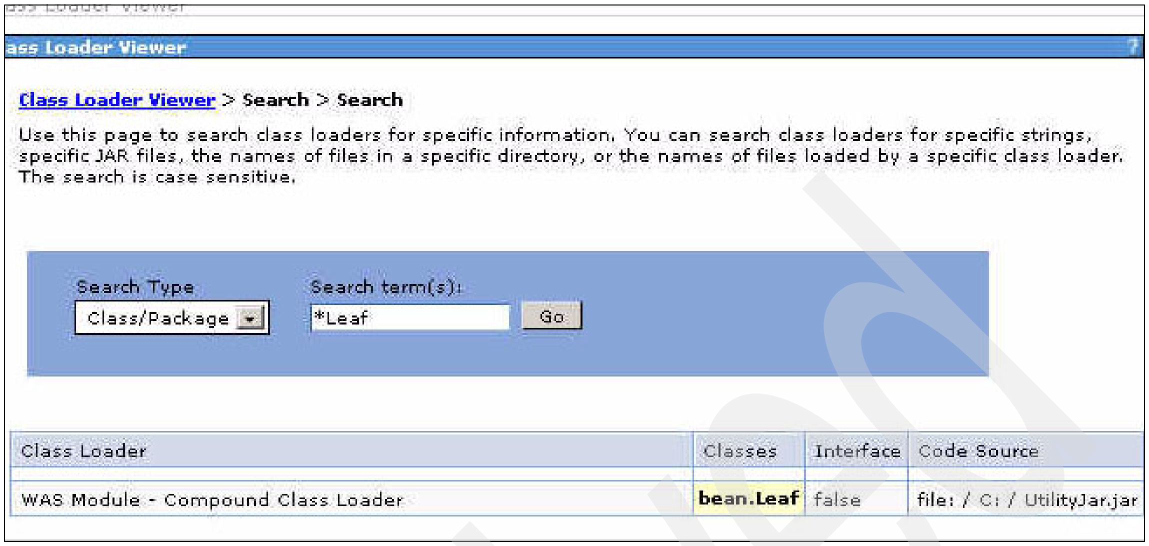
WAS Module - Compound Class Loader	
Delegation	true
Classpath	file:/C:/IBM/WebSphere/AppServer_v61/profiles/AppSrv01/installedApps/jas
	file:/C:/IBM/WebSphere/AppServer_v61/profiles/AppSrv01/installedApps/jas
	file:/C:/UtilityJar.jar
Classes	com.sun.faces.config.ConfigureListener

Figure 4-12   Table mode for the Class Loader Viewer

To use the search function, ensure that you enabled the Class Loader Viewer service, and select the **Search** option. This allows you to search for loaded classes.

Tip: Ensure that the class is loaded by invoking the application that will reference the class or else the Search option will not work.

Figure 4-13 on page 211 shows an example using `*Leaf` as the search parameter. The Class Loader Viewer finds classes with this pattern if they are loaded.


Class Loader Viewer

Class Loader Viewer > Search > Search

Use this page to search class loaders for specific information. You can search class loaders for specific strings, specific JAR files, the names of files in a specific directory, or the names of files loaded by a specific class loader. The search is case sensitive.

Search Type: Class/Package

Search term(s): *Leaf

Class Loader	Classes	Interface	Code Source
WAS Module - Compound Class Loader	bean.Leaf	false	file: / C: / UtilityJar.jar

Figure 4-13 Using the search functionality in Class Loader Viewer

4.11.3 Additional class loader diagnostics

JVM Version 5.0 provides some additional options that can be useful when troubleshooting class loading problems. These options are set as command line arguments for the JVM.

Select **Servers** → **Application Servers** → ***server_name*** and then expand the **Java and process management** section under the Server Infrastructure heading. Click the **Process Definition** link and then the **Java Virtual Machine** link. Enter the following options in the **Generic JVM arguments** field:

- -verbose:dynload

  This option provides detailed information about classes being loaded. Information includes the class name and package, the JAR file name if the class is packaged in a JAR file, the size of the class, and the time it takes to load the class. The information is written to the native_stderr.log file. An example output looks similar to the following:

  ```
  <Loaded servlet/LoadClass>
  <  Class size 1508; ROM size 1800; debug size 0>
  <  Read time 0 usec; Load time 40 usec; Translate time 32 usec>
  ```

- -Dibm.cl.verbose=*name*

  This option allows you to trace the way the class loaders find and load a given class. The name is the full name of the class, including the package name. By specifying -Dibm.cl.verbose=servlet.LoadClass, the following output is printed to the SystemOut log file when the LoadClass class is loaded.

Example 4-20 Output printed to SystemOutlogfile when LoadClass is loaded

```
[23/04/07 15:25:19:984 EDT] 00000024 SystemOut     O ExtClassLoader
attempting to find servlet.LoadClass
[23/04/07 15:25:19:984 EDT] 00000024 SystemOut     O ExtClassLoader
using classpath
C:\IBM\WebSphere\AppServer_v61\java\jre\lib\ext\CmpCrmf.jar;...
[23/04/07 15:25:19:984 EDT] 00000024 SystemOut     O ExtClassLoader
could not find servlet/LoadClass.class in
C:\IBM\WebSphere\AppServer_v61\java\jre\lib\ext\CmpCrmf.jar
...
...
[23/04/07 15:25:19:984 EDT] 00000024 SystemOut     O ExtClassLoader
could not find servlet.LoadClass

[23/04/07 15:25:19:984 EDT] 00000024 SystemOut     O AppClassLoader
attempting to find servlet.LoadClass
[23/04/07 15:25:19:984 EDT] 00000024 SystemOut     O AppClassLoader
using classpath
C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\properties;...
[23/04/07 15:25:20:000 EDT] 00000024 SystemOut     O AppClassLoader
could not find servlet/LoadClass.class in
C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\properties
...
...
[23/04/07 15:25:20:000 EDT] 00000024 SystemOut     O AppClassLoader
could not find servlet.LoadClass

[23/04/07 15:25:20:000 EDT] 00000024 SystemOut     O
com.ibm.ws.bootstrap.ExtClassLoader attempting to find
servlet.LoadClass
[23/04/07 15:25:20:000 EDT] 00000024 SystemOut     O
com.ibm.ws.bootstrap.ExtClassLoader using classpath
C:\IBM\WebSphere\AppServer_v61\java\lib;....
...
...
[23/04/07 15:25:20:015 EDT] 00000024 SystemOut     O
com.ibm.ws.bootstrap.ExtClassLoader could not find
servlet/LoadClass.class in C:\IBM\WebSphere\AppServer_v61\java\lib
...
...
```

Note that the output in Example 4-20 on page 212 was truncated to fit.

4.12 The next step

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, there are other class loader-related problems that you can experience.

4.12.1 Search online support

If you are sure the problem is with the class loader, there are things that you can do before contacting IBM support. First, you should review the documentation that you gathered for errors that were not addressed in this paper, and search support sites for information or fixes. Look for current information available from IBM support on known issues and resolutions on the following IBM support page:

`http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCVS24`

Look also at the WebSphere Information Center documentation for additional resources for diagnosing and fixing class loader issues:

- *Troubleshooting Class loaders*
  - Network Deployment on distributed platforms:

    `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/ttrb_classload_viewer.html`
  - Network Deployment on i5/OS

    `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.iseries.doc/info/iseriesnd/ae/ttrb_classload_viewer.html`

4.12.2 On-line resources

If, after going through this process, you still have an undiagnosed problem, we recommend that you return to *Approach to Problem Determination in WebSphere Application Server V6* at the following Web location:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf`

Review the problem classifications to see if there are any other components that might be causing the problem.

You can also review the troubleshooting process for class loaders available at the following Web location:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21242692

For more information about class loaders please refer to Chapter 12. Understanding class loaders in the IBM Redbooks publication *WebSphere Application Server V6.1: System Management and Configuration* at the following Web site:

http://www.redbooks.ibm.com/redbooks/pdfs/sg247304.pdf

4.12.3 Contact IBM

If these steps do not resolve your problem, then gather additional information as specified in the following MustGather document and raise a problem record with IBM. The following Web site contains a list of the MustGather documentation for Classloaders.

http://www-1.ibm.com/support/docview.wss?uid=swg21196187

4.13 Installing and configuring the sample application

The sample used in this paper is available for download from the following Web address:

ftp://www.redbooks.ibm.com/redbooks/REDP4307/

This is a sample class loader application that demonstrates some of the common class loader problems. Use the following instructions to install the application and to duplicate the exceptions.

The ClassloaderSample.zip will contain the following three files.

- ClassloaderSampleEAR.ear

  The enterprise application to be deployed to WebSphere Application Server V6.1.
- UtilityJar.jar

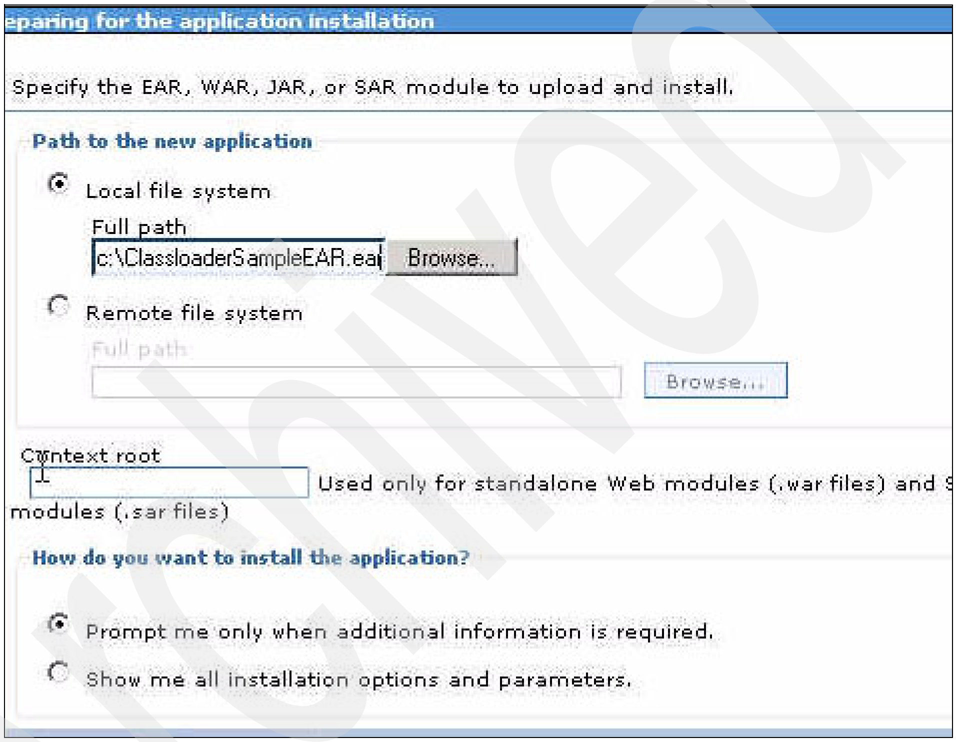
  Contains the utility classes for the ClassCastExceptionWeb module.
- MissingMethodUtilityJar.jar

  Contains the utility classes with a missing method for the NoSuchMethodError module.

Install the application

To install the application, do the following:

1. Unzip the zip file to a directory on your local system. For example C:\ for Windows.
2. Start the application server.
3. In the administrative console, expand **Enterprise Application** → **Install New Application**. Specify the location of the EAR as in Figure 4-14.



eparing for the application installation

Specify the EAR, WAR, JAR, or SAR module to upload and install.

Path to the new application

(●) Local file system

Full path: c:\ClassloaderSampleEAR.ear [Browse...]

( ) Remote file system

Full path: [Browse...]

Context root: Used only for standalone Web modules (.war files) and S modules (.sar files)

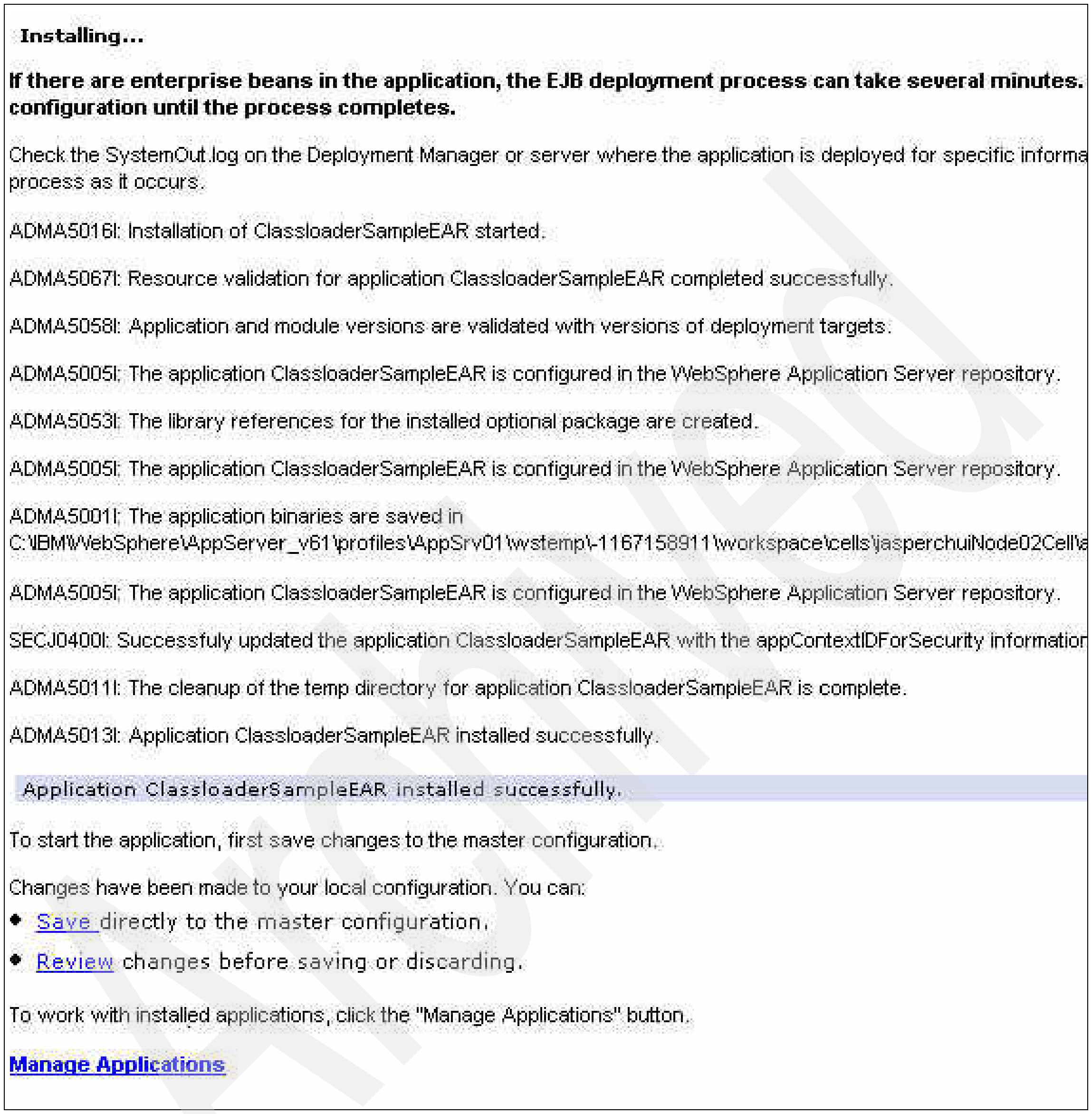
How do you want to install the application?

(●) Prompt me only when additional information is required.

( ) Show me all installation options and parameters.

Figure 4-14 Installing the sample class loader application

4. Step through the rest of the wizard, taking the default settings.
5. Click **Finish** to complete the install and to save the configuration. Figure 4-15 on page 216 shows the results of the installation.


Installing...

If there are enterprise beans in the application, the EJB deployment process can take several minutes. configuration until the process completes.

Check the SystemOut.log on the Deployment Manager or server where the application is deployed for specific informa process as it occurs.

```
ADMA5016I: Installation of ClassloaderSampleEAR started.

ADMA5067I: Resource validation for application ClassloaderSampleEAR completed successfully.

ADMA5058I: Application and module versions are validated with versions of deployment targets.

ADMA5005I: The application ClassloaderSampleEAR is configured in the WebSphere Application Server repository.

ADMA5053I: The library references for the installed optional package are created.

ADMA5005I: The application ClassloaderSampleEAR is configured in the WebSphere Application Server repository.

ADMA5001I: The application binaries are saved in
C:\IBM\WebSphere\AppServer_v61\profiles\AppSrv01\wstemp\-1167158911\workspace\cells\jasperchuiNode02Cell\a

ADMA5005I: The application ClassloaderSampleEAR is configured in the WebSphere Application Server repository.

SECJ0400I: Successfuly updated the application ClassloaderSampleEAR with the appContextIDForSecurity informatior

ADMA5011I: The cleanup of the temp directory for application ClassloaderSampleEAR is complete.

ADMA5013I: Application ClassloaderSampleEAR installed successfully.
```

Application ClassloaderSampleEAR installed successfully.

To start the application, first save changes to the master configuration.

Changes have been made to your local configuration. You can:
- Save directly to the master configuration.
- Review changes before saving or discarding.

To work with installed applications, click the "Manage Applications" button.

Manage Applications

Figure 4-15 Save the server configuration

Create the shared libraries

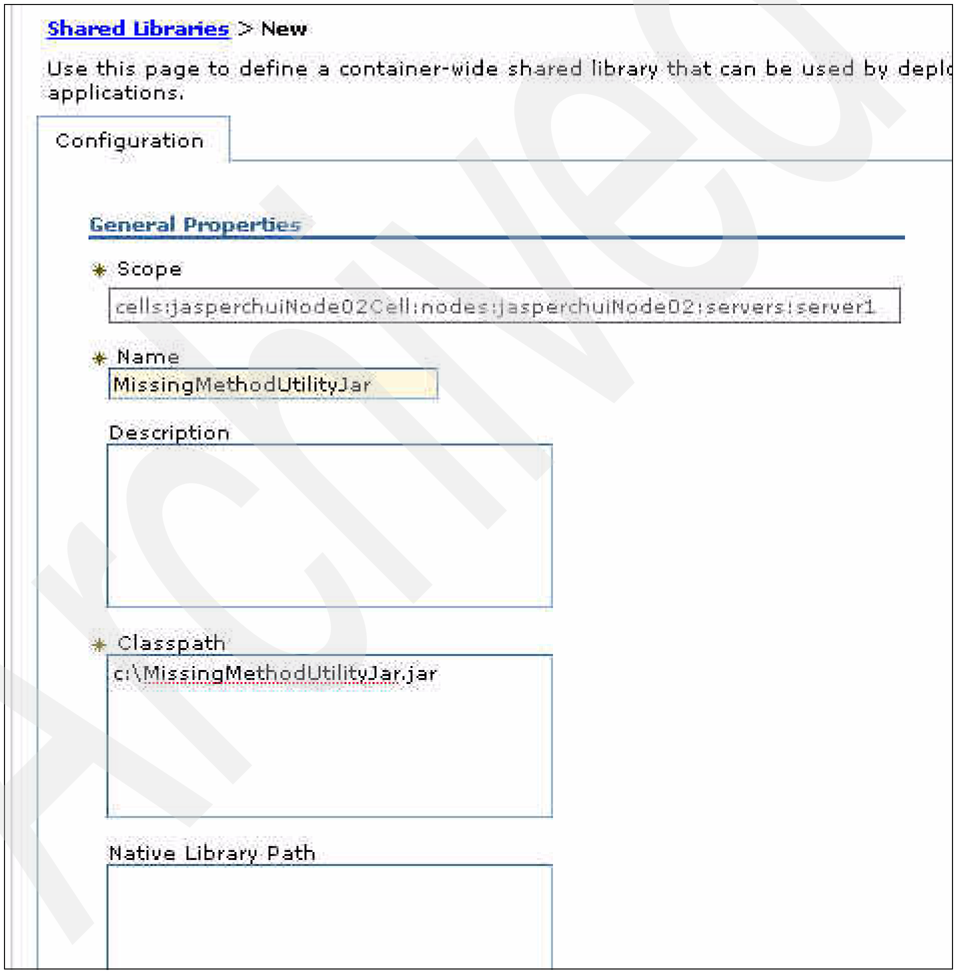
Create two shared libraries using the following steps.

1. In the administrative console, navigate to **Environment** → **Shared Libraries**.
2. Select **New** and specify the following:

- Name: `UtilityJar`
- Classpath: The location of the file (for example, `C:\UtilityJar.jar`).

3. Create another shared library for the MissingMethodUtilityJar.jar. Specify the following:
   - Name: `MissingMethodUtilityJar`
   - Classpath: The location of the file (for example,. `C:\MissingMethodUtilityJar.jar`).

   See Figure 4-16.



Shared Libraries > New

Use this page to define a container-wide shared library that can be used by deplo
applications.

Configuration

General Properties

* Scope
cells:jasperchuiNode02Cell:nodes:jasperchuiNode02:servers:server1

* Name
MissingMethodUtilityJar

Description

* Classpath
c:\MissingMethodUtilityJar.jar

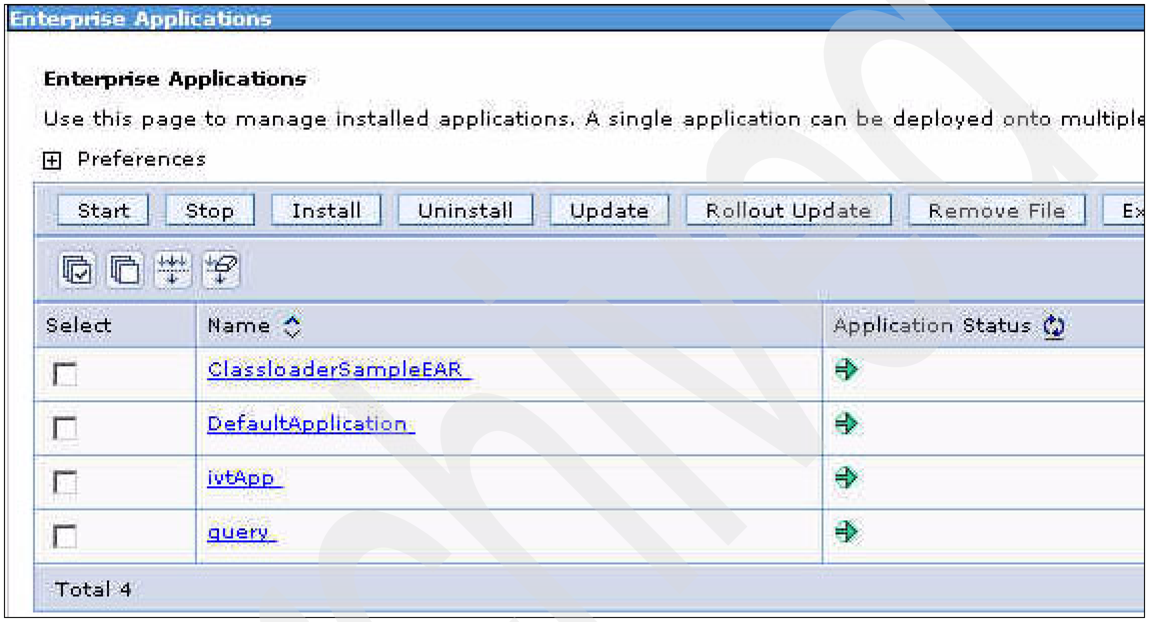
Native Library Path

Figure 4-16 Configuring MissingMethodUtilityJar.jar as a shared library

Start the application

To start the application, do the following:

1. In the administrative console navigate to **Enterprise Application** → **Enterprise Application**.
2. Ensure that the ClassloaderSampleEAR application is started. If it is not, select the check box located beside the application, and select **Start.** (Figure 4-17).



Enterprise Applications

Enterprise Applications

Use this page to manage installed applications. A single application can be deployed onto multiple

⊞ Preferences

Select	Name	Application Status
☐	ClassloaderSampleEAR	➜
☐	DefaultApplication	➜
☐	ivtApp	➜
☐	query	➜

Total 4

Figure 4-17   Starting the application

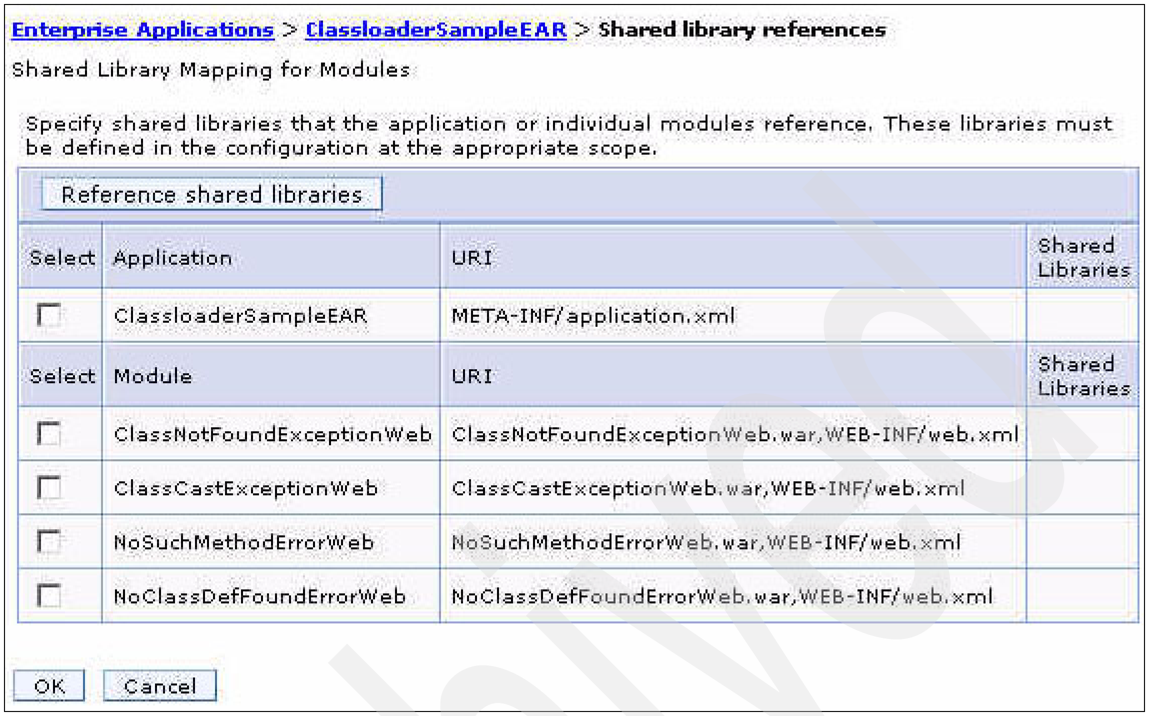
Configure the shared libraries for the application

To configure the shared libraries, do the following:

1. Click the ClassloaderSampleEAR application.
2. Select **Shared library references** under the References section.

   Figure 4-18 on page 219 shows the options to configure shared libraries and to associate them with a particular application or Web module. In this case, you will configure the shared library with the individual Web modules.

   Refer to for image of the initial configuration**.**


Enterprise Applications > ClassloaderSampleEAR > Shared library references

Shared Library Mapping for Modules

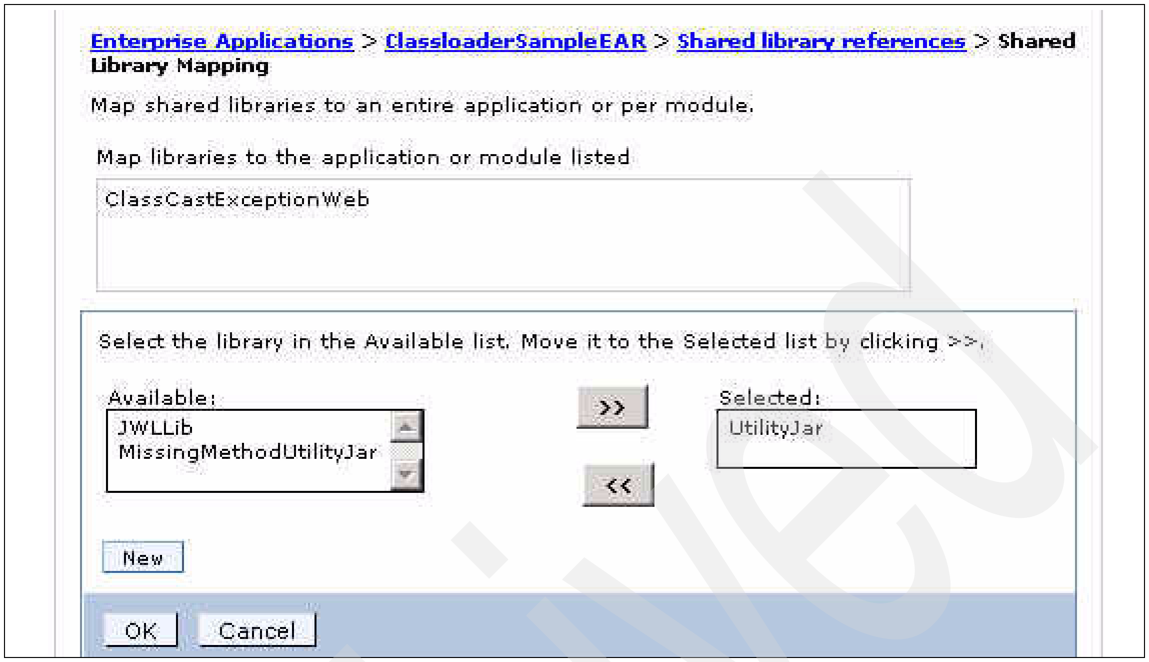
Specify shared libraries that the application or individual modules reference. These libraries must be defined in the configuration at the appropriate scope.

Reference shared libraries

Select	Application	URI	Shared Libraries
☐	ClassloaderSampleEAR	META-INF/application.xml	
Select	**Module**	**URI**	**Shared Libraries**
☐	ClassNotFoundExceptionWeb	ClassNotFoundExceptionWeb.war,WEB-INF/web.xml	
☐	ClassCastExceptionWeb	ClassCastExceptionWeb.war,WEB-INF/web.xml	
☐	NoSuchMethodErrorWeb	NoSuchMethodErrorWeb.war,WEB-INF/web.xml	
☐	NoClassDefFoundErrorWeb	NoClassDefFoundErrorWeb.war,WEB-INF/web.xml	

Figure 4-18 Associate shared libraries with the Web modules or the application

3. Select **ClassCastExceptionWeb**, and click **Reference shared libraries**.
4. Select the **UtilityJar** shared library, and click the arrow to associate it with this Web module. See Figure 4-19 on page 220 for the final configuration.


Enterprise Applications > ClassloaderSampleEAR > Shared library references > Shared Library Mapping

Map shared libraries to an entire application or per module.

Map libraries to the application or module listed

ClassCastExceptionWeb

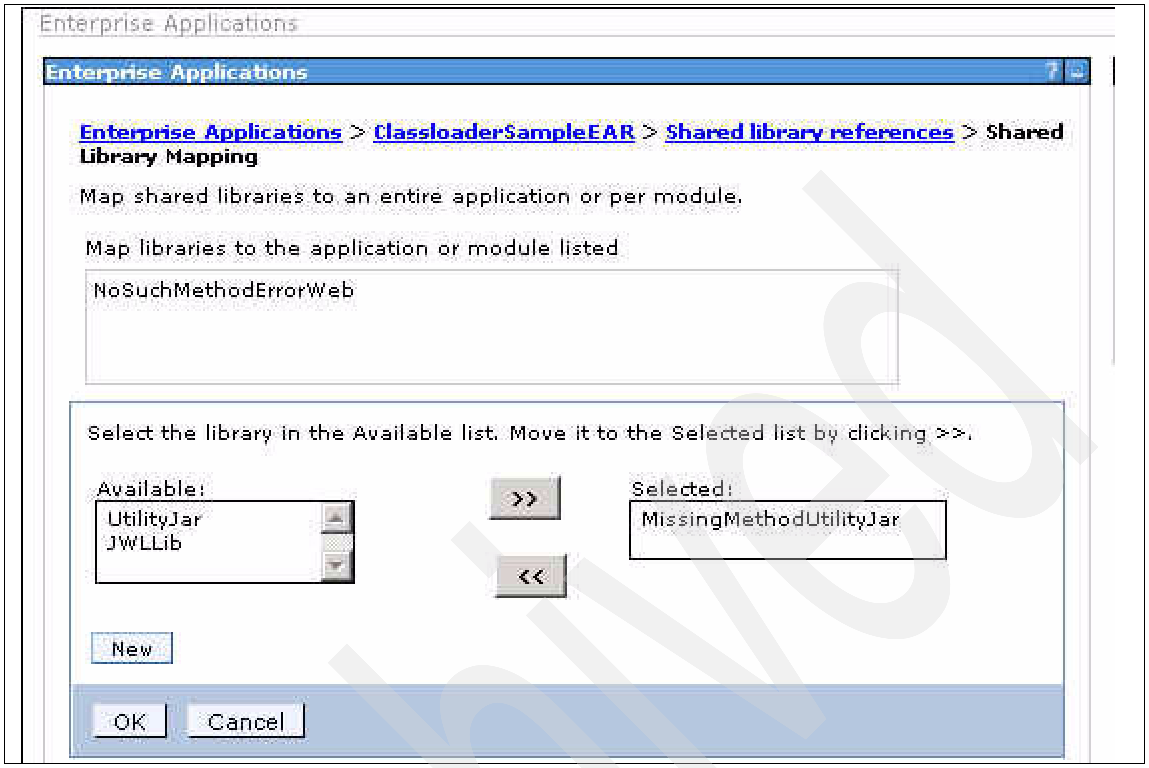
Select the library in the Available list. Move it to the Selected list by clicking >>.

Available:
JWLLib
MissingMethodUtilityJar

Selected:
UtilityJar

Figure 4-19 Associating UtillityJar shared library with ClassCastExceptionWeb web module.

5. Click **OK**.
6. Select **NoSuchMethodErrorWeb**, and click the **Reference shared libraries**.
7. Select the **MissingMethodUtilityJar** shared library, and click the arrow to associate it with this Web module. See Figure 4-20 on page 221 for the final configuration.


Enterprise Applications

Enterprise Applications

Enterprise Applications > ClassloaderSampleEAR > Shared library references > Shared Library Mapping

Map shared libraries to an entire application or per module.

Map libraries to the application or module listed

NoSuchMethodErrorWeb

Select the library in the Available list. Move it to the Selected list by clicking >>.

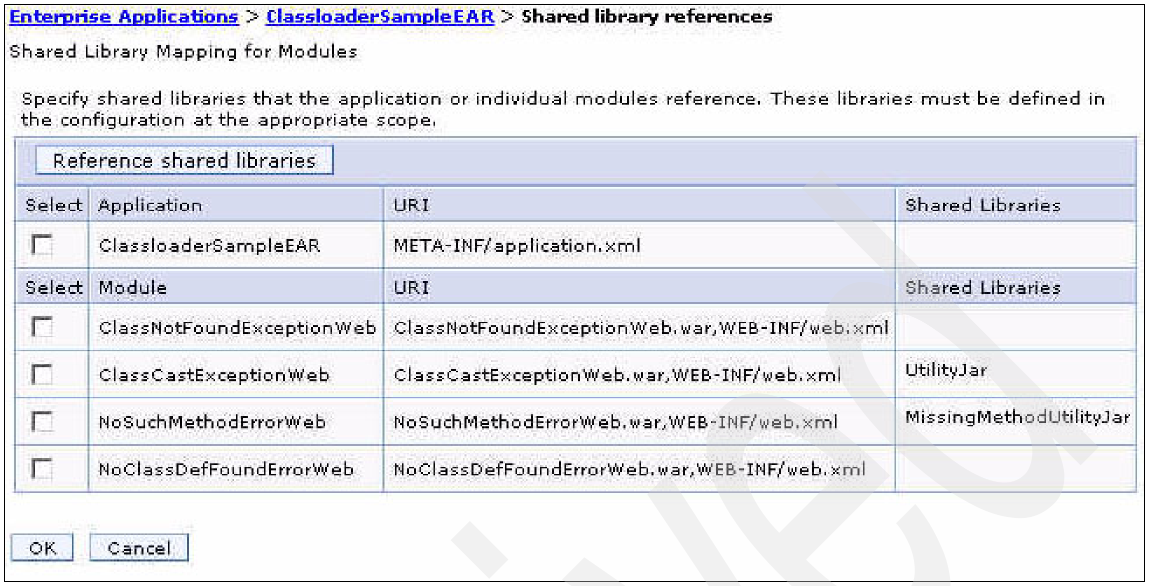
Available:
UtilityJar
JWLLib

Selected:
MissingMethodUtilityJar

Figure 4-20 Associating MissingMethodUtilityJar shared library with NoSuchMethodErrorWeb web module.

8. Select **OK**.

   The shared library references should be similar to Figure 4-21 on page 222.


Enterprise Applications > ClassloaderSampleEAR > **Shared library references**

Shared Library Mapping for Modules

Specify shared libraries that the application or individual modules reference. These libraries must be defined in the configuration at the appropriate scope.

Select	Application	URI	Shared Libraries
☐	ClassloaderSampleEAR	META-INF/application.xml	
Select	**Module**	**URI**	**Shared Libraries**
☐	ClassNotFoundExceptionWeb	ClassNotFoundExceptionWeb.war,WEB-INF/web.xml	
☐	ClassCastExceptionWeb	ClassCastExceptionWeb.war,WEB-INF/web.xml	UtilityJar
☐	NoSuchMethodErrorWeb	NoSuchMethodErrorWeb.war,WEB-INF/web.xml	MissingMethodUtilityJar
☐	NoClassDefFoundErrorWeb	NoClassDefFoundErrorWeb.war,WEB-INF/web.xml	

Figure 4-21 Final shared library configuration

9. Click **OK** to complete this action.
10. Select the option to **Save to the master configuration** of the server.

Produce the exceptions

Ensure that the application is started. From a Web browser, invoke the servlet to reproduce each exception.

- NoClassDefFoundError:
  - http://*hostname:port*/NoClassDefFoundErrorWeb/LoadClass
- ClassCastException
  - http://*hostname:port*/ClassCastExceptionWeb/LoadClass
- NoSuchMethodError
  - http://*hostname:port*/NoSuchMethodErrorWeb/LoadClass
- ClassNotFoundException
  - http://*hostname:port*/ClassNotFoundExceptionWeb/LoadClass

Where the *hostname* is the hostname of the machine and *port* is the defaulthost port for the application server profile.

Part 5

Workload managment and high availability problem determination

This part contains the following IBM Redpaper:

- *WebSphere Application Server V6.1: Workload Management Problem Determination*, REDP-4308 by Craig Scott

5

Load balancing problem determination

Load balancing problems with WebSphere Application Server can occur at various stages of the process.

This paper covers the following:

- Workload management problems with the Web server plug-in
- EJB workload management problems
- High availability manager problems

This chapter is applicable to load balancing problems that occur on WebSphere Application Server V6.1 on distributed and i5/OS platforms.

5.1 Introduction to workload management

Workload management (WLM) is a WebSphere facility that provides load balancing and affinity between application servers in a WebSphere clustered environment. WLM is an important facet of performance. WebSphere uses workload management to send requests to alternate members of the cluster. WebSphere can also be configured to route concurrent requests from a user to the application server that serviced the first request. This is called session affinity and can be used to maintain a user's session over concurrent HTTP requests.

WLM is configurable. The administrator should ensure that each machine or server in the configuration processes a fair share of the overall client load that is being processed by the system as a whole. Workload should be spread among machines such that the workload corresponds to the machine processing power.

5.1.1 WebSphere workload management

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS as shown in Figure 5-1.

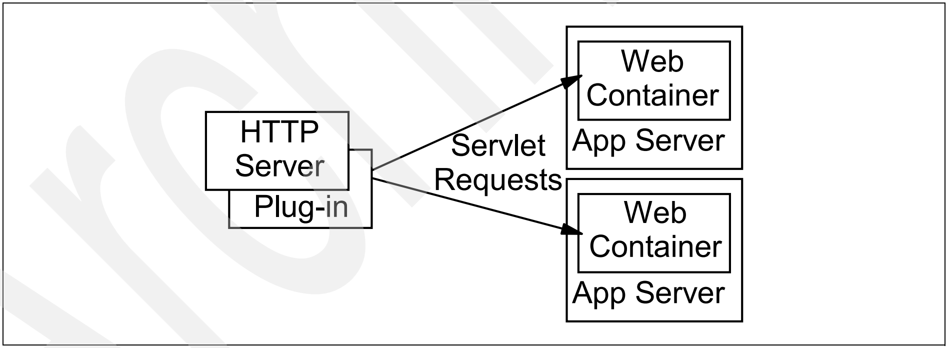


Figure 5-1 Plug-in (Web container) workload management

This routing is based on weights that are associated with the cluster members. If all cluster members have identical weights, the plug-in sends an equal number of requests to all members of the cluster, assuming no strong affinity configurations. If the weights are scaled in the range from zero to 20, the plug-in routes requests to those cluster members with the higher weight value more often. No requests are sent to cluster members with a weight of zero. Weights can be changed dynamically during runtime by the administrator.

The Web server plug-in temporarily routes around unavailable cluster members.

Multiple application servers with the EJB containers can be clustered, enabling the distribution of EJB requests between the EJB containers as shown in Figure 5-2.

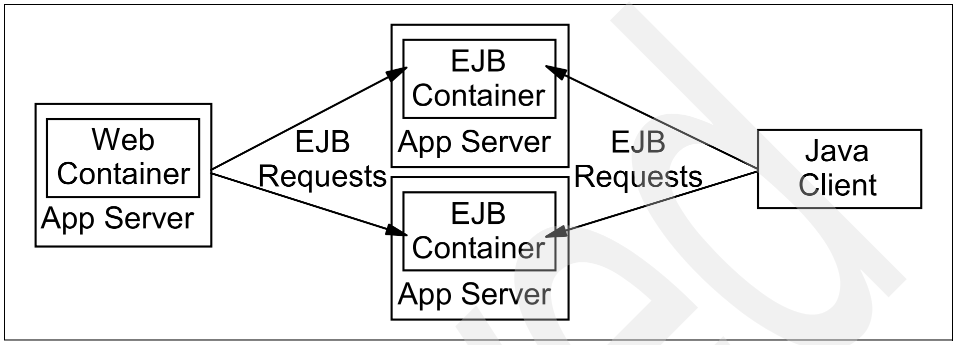


Figure 5-2 EJB workload management

In this configuration, EJB client requests are routed to available EJB containers in a round-robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

The server-weighted round-robin routing policy ensures a distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster is that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism sends more requests to the higher weight value servers than the lower weight value servers. The policy ensures the desired distribution, based on the weights that are assigned to the cluster members.

You can also choose to have requests sent to the node on which the client resides as the preferred routing. In this case, only cluster members on that node are chosen (using the round-robin weight method). Cluster members on remote nodes are chosen only if a local server is not available.

5.1.2 High availability

High availability is a concept closely related to workload management. Using high availability features is designed to eliminate single point of failures in the system. When one machine or server becomes unavailable, the workload is shifted to another.

Because problems during failover can affect workload balancing and performance, this paper includes problem determination guidance for the high availability manager.

5.2 Identify symptoms of a workload management problem

The primary symptom of a workload management problem is degradation of performance or problems during failover.

Specific symptoms of a problem with the Web server plug-in include:

- Plug-in not load balancing as expected
- Plug-in not detecting a failed server
- Plug-in not recovering from a server outage
- Session affinity not working
- Sessions not failing over under error conditions

Specific symptoms of a problem with EJB workload management include:

- EJB application requests do not get serviced
- EJB requests are not distributed evenly or to all servers
- Failing server still receives EJB requests (failover fails)
- Restarted servers do not share the workload

The following are typical symptoms of an HA Manager problem.

- Singleton services not starting on failover
- Singleton services starting unexpectedly
- Server will not start
- Excessive resource usage
- HA Manager messages in the logs. These will be prefixed with HMGR, CWRLS or DCSV.

5.3 Determine problem type

Workload management problems can occur at various stages depending on your configuration. The following is an approach you can take to navigate through this paper and diagnose a workload management problem.

1. Are you using clustering?
   a. If not, then the activities in this Redpaper will not help you.
   b. If yes, go to step 2.
2. Start the problem determination process by verifying that the Web server plug-in is working correctly and spreading work across the application servers as intended. Chapter 6, "Web server plug-in load balancing problem determination" on page 233 can guide you through this process.

   If you determine that there is no problem with the Web server plug-in, continue to step 3.
3. Continue the process by verifying that requests for EJBs are being distributed among the application servers in the cluster as you intended. Chapter 7, "EJB workload management problem determination" on page 267 can guide you through this process.

   If you determine there is no problem with the EJB workload management continue to step 4.
4. Are you using the high availability manager? This is the default. Chapter 8, "High availability manager problem determination" on page 289 can guide you through the process of determining whether you have a problem in the High Availability Manager.
5. If, after going through this process, you still have an undiagnosed problem, we recommend that you go back to *Approach to Problem Determination in WebSphere Application Server V6* at:

   `http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf`

   Review the problem classifications to see if there are any other components that might be causing the problem.

6

Web server plug-in load balancing problem determination

The Web server plug-in handles communications between a Web server and the WebSphere Application Server. The plug-in can direct requests to multiple WebSphere Application Servers to provide load balancing and failover functionality. Problems with the plug-in can cause performance to degrade or processing to behave abnormally.

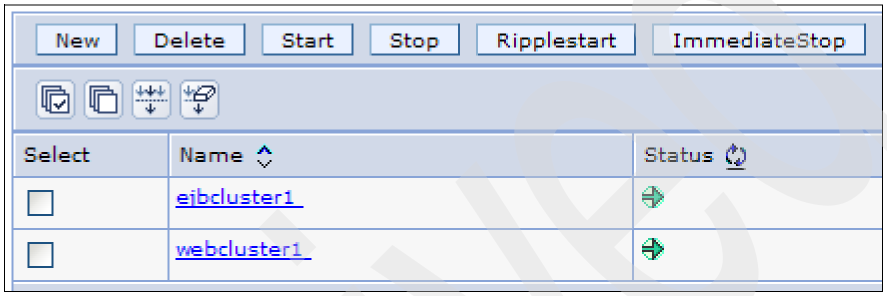
The plug-in provides load balancing to direct requests to each server in a WebSphere Application Server cluster in turn. It chooses which server to send a request to, based on either a weighted round robin algorithm or by random distribution. The plug-in also supports session affinity where an HTTP request is routed back to the same application server that originally created the HTTP session for that user.

The plug-in also provides failover capability, where one or more of the servers that it is directing requests to is shut down or “crashes”. The plug-in will detect that an application server is no longer responding to requests and will stop sending requests to that server. The plug-in will periodically check back with the server to see if it has become available again.

6.1 Check system integrity

If you think you might have a Web server plug-in WLM problem, the first step is to verify the system integrity:

- Verify that all cluster member servers are up and running. In the administrative console, navigate to **Servers** → **Clusters** and ensure your clusters are showing as started as in Figure 6-1. All cluster members need to be started to ensure even load balancing.



Select	Name	Status
☐	ejbcluster1	➔
☐	webcluster1	➔

Figure 6-1 Started clusters

- Ensure all cluster members are responding as expected. You can do this by using your Web browser to connect to each application server directly rather than through the HTTP server and plug-in.

  First determine the HTTP transport port number (or HTTPS transport port number if using SSL) and then access the application URL using the application server host name and port number from your Web browser

  You can determine the HTTP and HTTPS transport port number for each application server from a text file that is in each profile's log directory named `AboutThisProfile.txt`. This file will list the HTTP transport port number as shown in Example 6-1:

Example 6-1 AboutThisProfile.txt

```
Application server environment to create: Application server
Location: /opt/IBM/WebSphere/AppServer/profiles/Node2
Disk space required: 200 MB
Profile name: Node1
Make this profile the default: False
Node name: IBM99TVXRDNode1
Host name: IBM99TVXRD.au.ibm.com
Enable administrative security (recommended): False
Administrative console port: 9061
Administrative console secure port: 9044
```

```
HTTP transport port: 9081
HTTPS transport port: 9444
Bootstrap port: 2810
...
```

For example: Assume you normally access your application via an external URL such as `www.ibm.com`. This URL points to a Web server with a WebSphere plug-in that load balances requests across the two servers named `appserver1` and `appserver2`. You would normally connect to your application using a URL like the following:

`http://www.ibm.com/wlm/BeenThere`

After determining the HTTP transport port number of the application server running on `appserver2` to be `9081`, you would access the application from your browser via:

`http://appserver2:9081/wlm/BeenThere`

- Check your plug-in fix pack level

  The plug-in fix pack level must be equal to or higher than all of the application servers that it is routing requests to. If you have applied fix packs to your application servers, ensure you have also applied the equivalent fix packs to your plug-in. You can check the fixpack levels of WebSphere Application Server and the WebSphere plug-in using the `versionInfo.bat` or `versionInfo.sh` script that is provided in the **`bin`** directory of each installation.

  See the following URL for more information about supported combinations of Web server plug-in and WebSphere Application Server:

  – Web server plug-in policy for WebSphere Application Server

    http://www-1.ibm.com/support/docview.wss?uid=swg21160581

Tip: Your deployment manager must also be at a higher fix pack level than the application servers it manages. It is simplest to keep all components at the same fix pack level.

6.1.1 Identify symptoms

The following are typical symptoms of a plug-in WLM problem.

- Plug-in not load balancing as expected
- Plug-in not detecting a failed server
- Plug-in not recovering from a server outage

- Session affinity not working
- Sessions not failing over under error conditions

6.2 Plug-in not load balancing as expected

This section discusses steps to diagnose your problem if you have noticed that application requests are not being distributed equally to all application servers in the cluster

6.2.1 Collect diagnostics

The most important source of diagnostic data for analyzing plug-in load balancing problems is the plug-in log.

You should also review PMI data from each application server to monitor the number of requests being served and also server host metrics to ensure servers are not being overloaded.

You may also need to review logs from all the application servers in the cluster to determine if a particular cluster member is experiencing some problem that prevents it from servicing requests.

Collect the following:

- Plug-in log

  See 9.3, “Collecting the plug-in log” on page 317.
- Application server logs:
  - JVM SystemOut and SystemErr logs

    See 9.1, “Collecting JVM logs” on page 314.
  - native_stderr.log

    By default, this file is located in *profile_root*/logs/*server_name*/native_stderr.log

Depending on what you find, you may also need to look at the following:

- PMI data
- Server performance metrics

6.2.2 Analyze diagnostics

You will first need to review the plug-in log to determine if a server has been marked down.

Analyze the plug-in log

Look for the following in http-plugin.log

- Messages to indicate that a server has been marked down and that the plug-in is no longer sending it requests. Example 6-2 shows an application server being marked down as the plug-in is no longer able to connect to server.

Example 6-2 Server marked down

```
[Thu Apr 12 15:20:13 2007] 00001288 00001f90 - ERROR: ws_common:
websphereGetStream: Failed to connect to app server on host
'IBM99TVXRD.au.ibm.com', OS err=10061
[Thu Apr 12 15:20:13 2007] 00001288 00001f90 - ERROR: ws_common:
websphereExecute: Failed to create the stream
[Thu Apr 12 15:20:13 2007] 00001288 00001f90 - ERROR: ws_server:
serverSetFailoverStatus: Marking IBM99TVXRDNode1_server1 down
[Thu Apr 12 15:20:13 2007] 00001288 00001f90 - ERROR: ws_common:
websphereHandleRequest: Failed to execute the transaction to
'IBM99TVXRDNode1_server1'on host 'IBM99TVXRD.au.ibm.com'; will try another one
```

  If you find a server is marked down, restart the server and then go to "Validate the solution" on page 246.

  If you find an application server is being marked down, but you determine the server is not actually down, go to "Server being marked down unexpectedly" on page 246.

  If this is not the cause of the problem, then you will need to determine the load balance distribution to investigate further by enabling plug-in detail logging and by reviewing PMI data.

Analyze the plug-in detail logging

To verify plug-in load balancing, you will need to enable further logging at the plug-in. The default level of logging is Error. Increase your plug-in log level and then run a test to recreate the issue. If you have a simple environment with only one clustered application, you can use the `Stats` logging level. If your environment is complex, use the `Trace` logging level. See 9.3.1, "Setting the log level" on page 317 for more information on changing the plug-in log tracing level.

Look for the following:

- Using the `Warn` logging level, you may see the following message:

```
[Fri May 04 11:39:10 2007] 000000d8 000013e0 - WARNING:
ws_server_group: serverGroupCheckServerStatus: Server
IBM99TVXRDNode2_server2 has reached maximmum connections and is not
selected
```

Go to “Max connections for the server has been reached” on page 244.

- Using the `Stats` logging level, you will see output as shown in Example 6-3. This is showing the distribution of requests across the application servers. However `Stats` does not show the different URI requests. When you have multiple applications in your environment, you will not be able to determine the distribution of requests for the different applications.

Example 6-3 Stats output

```
[Thu Apr 12 14:31:29 2007] 000018f4 00001b70 - STATS: ws_server:
serverSetFailoverStatus: Server IBM99TVXRDNode2_server2 : pendingRequests 0
failedRequests 0 affinityRequests 0 totalRequests 54.
[Thu Apr 12 14:31:33 2007] 000018f4 00001b70 - STATS: ws_server:
serverSetFailoverStatus: Server IBM99TVXRDNode1_server1 : pendingRequests 0
failedRequests 0 affinityRequests 0 totalRequests 55.
```

- Using the **`Trace`** log level, you will get sufficient information to determine the distribution of requests based on the applications running in your environment. However, the amount of data generated will be large as it will log the sequence of events that occurs as each request that comes through the plug-in is processed. You should only run with this logging level when necessary.

  In the Web server plug-in configuration file, you can see that the context root from each installed application is mapped to an application server or cluster that can handle that request.

  To process the results from the http-plugin.log file, you will need to look through the file and extract the data that shows request distribution. Example 6-4, shows extracts from a plug-in `Trace` of a request to the `snoop` servlet.

  You can see the request come in to the plug-in, the plug-in tells you it is using a round robin algorithm and that it has chosen **IBM99TVXRDNode2_server2** to process the request. Near the end of the request output, the plug-in prints the overall statistics for the chosen server.

Example 6-4 Extracting request distribution by URI

```
[Thu Apr 12 14:40:24 2007] 0000190c 000018e8 - DEBUG: lib_util:
parseHostHeader: Defaulting port for scheme 'http'
[Thu Apr 12 14:40:24 2007] 0000190c 000018e8 - DEBUG: lib_util:
parseHostHeader: Host: 'ibm99tvxrd', port 80
```

```
[Thu Apr 12 14:40:24 2007] 0000190c 000018e8 - DEBUG: ws_common:
websphereCheckConfig: Current time is 1176352824, next stat time is
1176352859
[Thu Apr 12 14:40:24 2007] 0000190c 000018e8 - DETAIL: ws_common:
websphereShouldHandleRequest: trying to match a route for:
vhost='ibm99tvxrd'; uri='/snoop'
...
Thu Apr 12 14:40:24 2007] 0000190c 000018e8 - DEBUG: ws_server_group:
serverGroupNextRoundRobinServer: Round Robin load balancing
...
[Thu Apr 12 14:40:24 2007] 0000190c 000018e8 - DEBUG: ws_server_group:
serverGroupNextRoundRobinServer: use server IBM99TVXRDNode2_server2
...
[Thu Apr 12 14:40:24 2007] 0000190c 000018e8 - STATS: ws_server:
serverSetFailoverStatus: Server IBM99TVXRDNode2_server2 :
pendingRequests 0 failedRequests 0 affinityRequests 0 totalRequests 85.
```

A single request to the snoop servlet at `Trace` log level will produce 187 lines of logging data in the `http-plugin.log`. In a production environment with high transaction rates, the amount of data to parse is going to get very large very quickly and it is not practical to process it by hand. For this reason you should consider using a scripting language such as Python or Perl to assist you process this data and generate summary reports.

If you confirm that you are experiencing uneven load balancing, then you will need to look further to determine why.

- Check the plug-in configuration to ensure the server weights are set as appropriate. For more information, go to “Server weights not equal” on page 245.
- If you are seeing a server not being selected, as the maximum number of requests has been reached, go to “Max connections for the server has been reached” on page 244.
- Plug-in logging levels `Stats` and higher will also report on requests that have been routed to maintain session affinity. If you are seeing uneven load balancing due to session affinity, go to “Session affinity is skewing load distribution” on page 245.

Review the application logs, PMI statistics and server metrics to determine why a server is not processing requests, or processing less requests than you expect.

Analyze the application server logs

For each application server that is not participating in the load balancing, look for exceptions or errors in the SystemOut or SystemErr log that might indicate why the application server is not servicing requests. Some of the problems you might find include:

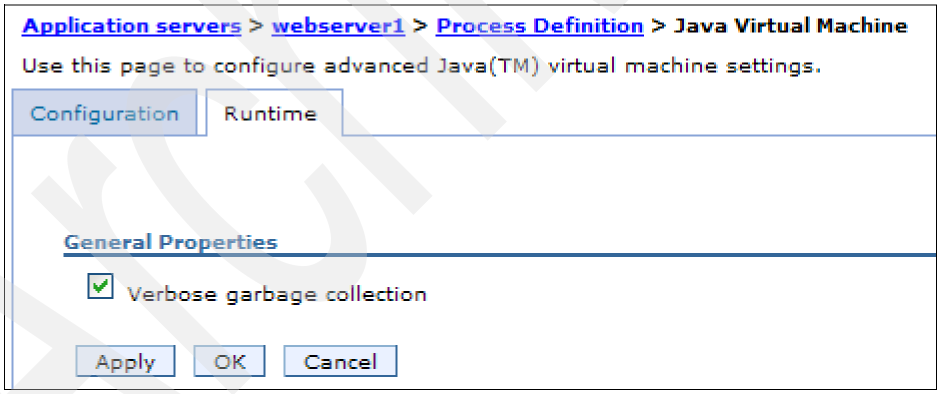
- The application server has crashed and is restarting
- The application has hung and is not processing requests
- The application server has run out of memory
- Other application related errors to indicate why requests cannot be processed. Note however that these will not typically cause the plug-in to mark the server down, as the server is still responding to requests, just with an error.

Note: The resolution of these issues is outside the scope of this paper.

- Check verbose GC data

  If your JVM heap size usage is getting close to its maximum, you are likely to see an excessive number of garbage collections being performed. Frequent garbage collections in the JVM will increase the CPU being used on the server and reduce the number of requests that can be processed.

  Verbose GC can be enabled dynamically on a running server through the administrative console, see Figure 6-2.


Application servers > webserver1 > Process Definition > Java Virtual Machine

Use this page to configure advanced Java(TM) virtual machine settings.

General Properties

[✓] Verbose garbage collection

Figure 6-2 Enable verbose GC in the admin console

If you find you are experiencing excessive Java heap usage, you will need to pursue that as the root cause. Java heap problems are outside the scope of this paper. For a list of references that can help you pursue this type of problem, see 6.7.1, “Java heap problems” on page 264.

- Look for an out of memory condition

  If your available JVM heap size or process native memory has been exhausted, you will see an out of memory exception reported in the logs.

  In the case of JVM heap size exhaustion, you will see excessive garbage collection activity leading up to this point. When the memory is exhausted, the server will report the condition and generate a javacore and heapdump file for your analysis. The server may continue to function or it may crash depending on the circumstances. The JVM may continue to try and service requests.

  In the case of native memory exhaustion, the JVM will crash and will generate a javacore, core file or user.dmp file. The JVM will no longer participate in servicing requests.

  If you find you are experiencing excessive Java heap usage, you will need to pursue that as the root cause. Java heap problems are outside the scope of this paper. For a list of references that can help you pursue this type of problem, see 6.7.1, "Java heap problems" on page 264.

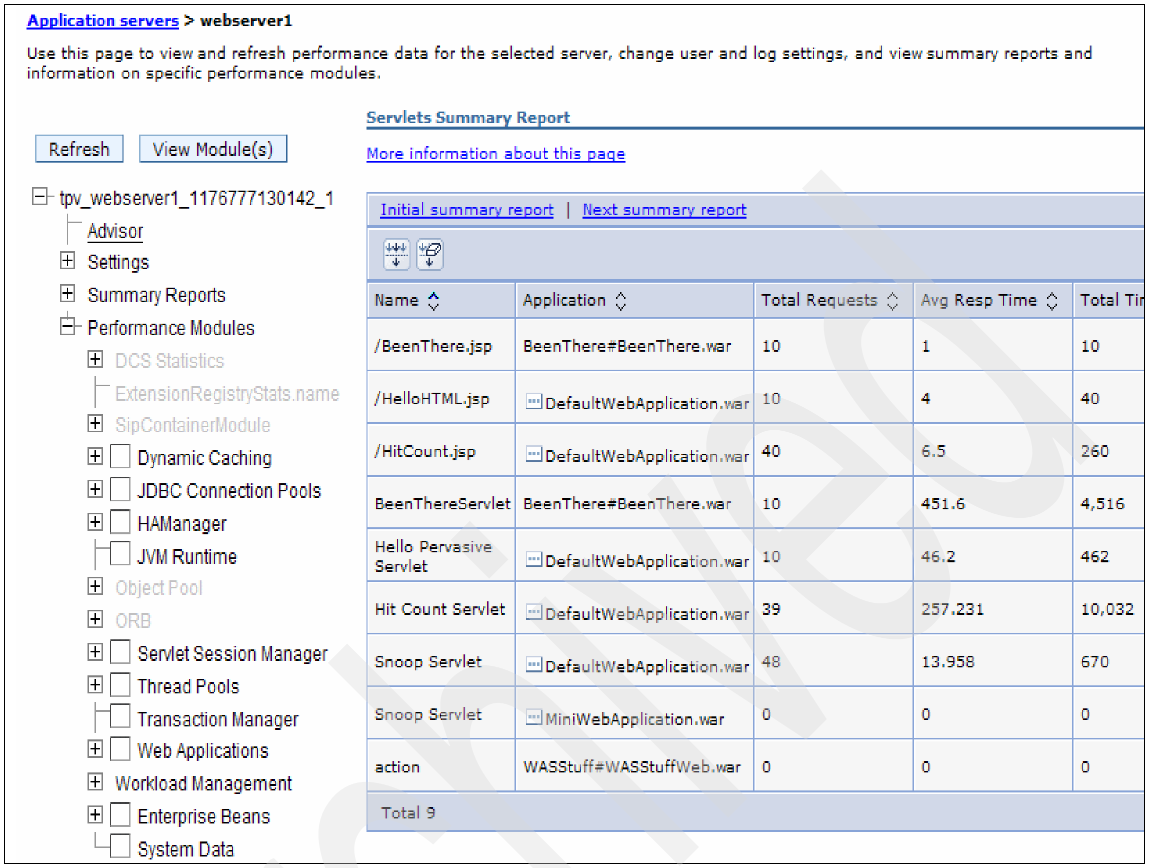
Performance Monitoring Infrastructure (PMI) data

Check the PMI data to ensure all cluster members are processing the expected number of requests and there are no adverse statistics such as high CPU or JVM memory usage that might indicate a problem.

Look for the following:

- You can review the number of requests being processed by each application server using the Tivoli® Performance Viewer (TPV) or other PMI client. TPV will also show other problems such as excessive heap usage or garbage collection.

  Navigate to **`Monitoring and tuning`** → **`Performance viewer`** → **`Current activity`** and then click each server name in the cluster in turn. Figure 6-3 shows the initial view you will see when opening the viewer and will show you the number of requests serviced by each servlet or JavaServer Pages (JSP) and the average response time. Comparing these statistics across servers will show the distribution of load balanced requests from the application server point of view. Compare this to what you expect to see from the statistics gathered from the plug-in.


Application servers > webserver1

Use this page to view and refresh performance data for the selected server, change user and log settings, and view summary reports and information on specific performance modules.

Servlets Summary Report

More information about this page

Initial summary report | Next summary report

Name	Application	Total Requests	Avg Resp Time	Total Ti
/BeenThere.jsp	BeenThere#BeenThere.war	10	1	10
/HelloHTML.jsp	DefaultWebApplication.war	10	4	40
/HitCount.jsp	DefaultWebApplication.war	40	6.5	260
BeenThereServlet	BeenThere#BeenThere.war	10	451.6	4,516
Hello Pervasive Servlet	DefaultWebApplication.war	10	46.2	462
Hit Count Servlet	DefaultWebApplication.war	39	257.231	10,032
Snoop Servlet	DefaultWebApplication.war	48	13.958	670
Snoop Servlet	MiniWebApplication.war	0	0	0
action	WASStuff#WASStuffWeb.war	0	0	0

Total 9

Figure 6-3   PMI Statistics

If you find uneven processing of requests but there are no messages in the plug-in log to indicate why, you will need to increase the plug-in logging level to further debug this issue. Go to “Analyze the plug-in detail logging” on page 237.

- Review the JVM heap usage in the TPV to get an indication of the health of the application server. Navigate to **`Monitoring and tuning`** → **`Performance viewer`** → **`Current activity`** and then click each server name in the cluster in turn. Expand the **`Performance modules`** tree in the left hand side of the TPV and click **`JVM Runtime`**. Then click **`Show modules`**. Figure 6-4 is an example of a healthy JVM heap usage.

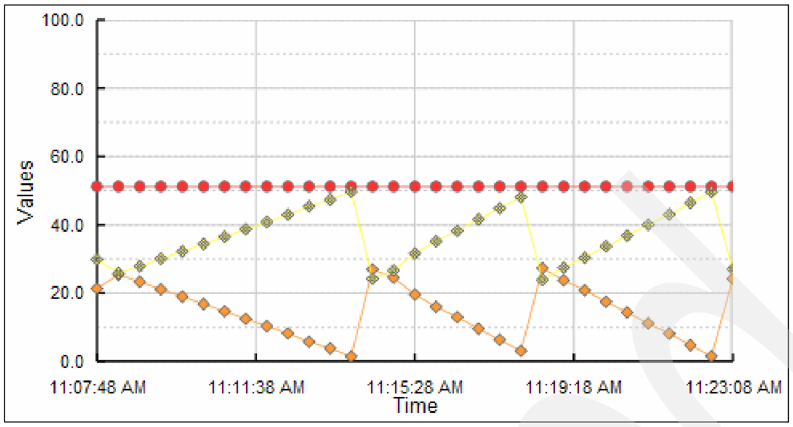


Figure 6-4   JVM heap usage

If you find you are experiencing excessive Java heap usage, you will need to pursue that as the root cause. Java heap problems are outside the scope of this paper. For a list of references that can help you pursue this type of problem, see 6.7.1, “Java heap problems” on page 264.

Server metrics

Server metrics such as CPU, I/O and page space utilization can also show uneven or unexpected load balancing. You should use the tool most appropriate to your operating system to check these statistics. For example, in Windows you could use `Performance Monitor`. In a Unix or Linux environment, you could use `vmstat` or `top`.

Look for the following:

- Unexpectedly high CPU, I/O or page space usage on one server compared to the others. This could be indicative of uneven load balancing. However it could also be indicative of other problems such as a non WebSphere related process using too much CPU, excessive garbage collection or a looping WebSphere Application Server.

  Resolution of these issues is outside the scope of this paper.

6.2.3 Root causes

The following are possible root causes for uneven load balancing:

- Max connections for the server has been reached
- Server weights not equal
- Session affinity is skewing load distribution

- Server being marked down unexpectedly

6.2.4 Max connections for the server has been reached

If your plug-in as been configured to restrict the number of connections that a server can accept (`maxConnections`), you may be reaching this maximum. Example 6-5 shows a server that has reached this limit with the logging level set to `Stats`.

Example 6-5 Max connections reached

```
[Fri Apr 20 15:49:11 2007] 000008d8 000000d0 - STATS: ws_server_group: 
serverGroupCheckServerStatus: Checking status of IBM99TVXRDNode2_server2, 
ignoreWeights 0, markedDown 0, retryNow 0, retryInSec --, wlbAllows 11 
reachedMaxConnectionsLimit 1
```

Resolve the problem

`maxConnections` is configured individually for each server in a cluster by editing the plug-in configuration file. You can determine the path to the configuration file through the WebSphere admin console. Navigate to **Web servers** → ***web_server_name*** → **Plug-in properties** and look for `Web server copy of Web server plug-in files`. You will find the path to the running copy of the Web server plug-in configuration file in the `Plug-in configuration directory and file name` box, for example:

```
/opt/IBM/WebSphere/IHS/Plugins/config/webserver/plugin-cfg.xml
```

Example 6-6 shows the default setting of the `MaxConnections` parameter on `server2` from the plug-in configuration file.

Example 6-6 Max connections configuration

```
<Server CloneID="1251sllkq" ConnectTimeout="0" ExtendedHandshake="false"
LoadBalanceWeight="2" MaxConnections="-1" Name="IBM99TVXRDNode2_server2"
ServerIOTimeout="0" WaitForContinue="false">
```

The default value of `MaxConnections` is “-1” which means no limit is imposed by the plug-in. If you find `MaxConnections` has been set to a specific value, you will need to determine why it was set from the system administrator who made this change. The value chosen may no longer be valid and should be reviewed.

Note: It is not possible to modify MaxConnections through the WebSphere administrative console.

6.2.5 Server weights not equal

In the configuration of the plug-in, you can set server weights that will adjust the ratio of requests that are directed to each server. For example, if you have two servers where one server is four times more powerful than the other, you might adjust server weights so the more powerful server services four requests to every one serviced by the other.

If you do this, you will not see even distribution of requests among your application servers by design. A more detailed discussion of this can be found at:

- Understanding IBM HTTP Server plug-in Load Balancing in a clustered environment

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219567

Resolve the problem

Adjust server weights as appropriate for each of the cluster member servers.

6.2.6 Session affinity is skewing load distribution

Session affinity can cause the plug-in to distribute requests unevenly.

Example 6-7 shows an imbalance in load balancing caused by session affinity. You can see that server1 has serviced 448 requests while server2 has only serviced 45 requests, a ratio of roughly 10 to 1. However you can also see that of the 448 requests serviced by server1, 405 of these were sent to that server to maintain session affinity. Without the session affinity requests, the load balance ratio is almost 1 to 1.

Example 6-7 http-plugin.log showing skewed load balancing due to session affinity

```
[Thu Apr 12 16:26:15 2007] 00001288 00001f90 - STATS: ws_server:
serverSetFailoverStatus: Server IBM99TVXRDNode1_server1 : pendingRequests 0
failedRequests 0 affinityRequests 405 totalRequests 448.
[Thu Apr 12 16:26:16 2007] 00001288 00001300 - STATS: ws_server:
serverSetFailoverStatus: Server IBM99TVXRDNode2_server2 : pendingRequests 0
failedRequests 0 affinityRequests 0 totalRequests 45.
```

Resolve the problem

Adjust server weights as per the following technote to try and balance server affinity based requests more appropriately.

- Understanding IBM HTTP Server plug-in Load Balancing in a clustered environment

  `http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219567`

This situation is often seen in test environments and this imbalance will probably sort itself out as the number of distinct concurrent users with unique sessions increases. As the number of unique sessions increases, the plug-in will be able to distribute these more evenly across the cluster members.

6.2.7 Server being marked down unexpectedly

In some instances, you may find that the plug-in is marking an application server down for no apparent reason. You can access the application on the server's HTTP transport port yet the plug-in log is reporting that the server has been marked down.

Example 6-2 on page 237 shows the error you will see in the plug-in log.

Resolve the problem

This could be caused by intermittent error conditions on the application server. If the application server is unable to process a connection for a particular request, the plug-in will mark the server down. This could be due to resource constraints such as insufficient memory to establish a TCP/IP connection or some other error condition. Check the application server logs.

You can also see this condition intermittently if the plug-in fix pack level is lower than the application server fix pack level. Apply the appropriate fix pack to your plug-ins.

6.2.8 Validate the solution

Recreate the situation and validate that requests are being distributed to the clustered servers as you expect.

If this does not resolve the issue, go to "The next step" on page 264.

6.3 Plug-in not detecting a failed server

This section discusses steps to diagnose the problem where an application server is no longer responding to requests, yet the plug-in is still attempting to send requests to that server. This situation will cause a delay in processing while the plug-in waits for the server to respond or for the TCP connection to time-out.

6.3.1 Collect diagnostics

You will need to review a plug-in trace to see why the plug-in is trying to send requests to a server that is not responding. You will also need to review the JVM logs from the application server that is failing or has failed.

- Plug-in log using the Trace logging level.

  See 9.3, "Collecting the plug-in log" on page 317.
- JVM SystemOut and SystemErr logs

  See 9.1, "Collecting JVM logs" on page 314.

6.3.2 Analyze diagnostics

You will first need to set the plug-in logging level to Trace, recreate the problem and review the trace data generated in the http-plugin.log

Analyze the plug-in trace

Look for the following:

- The plug-in establishing a connection to an application server and not getting a response.

  In Example 6-8, you can see the plug-in choosing a server, `server1`, and building the request stream to be passed to the application server. It creates the stream to the application server and sends the request.

Example 6-8 A request with no response

```
[Fri May 04 12:34:27 2007] 00001660 000019d4 - TRACE: ws_server_group:
serverGroupIncrementConnectionCount: Server IBM99TVXRDNode1_server1 picked,
pendingConnectionCount 2 totalConnectionsCount 2.
[Fri May 04 12:34:27 2007] 00001660 000019d4 - DEBUG: ws_server_group:
serverGroupNextRoundRobinServer: use server IBM99TVXRDNode1_server1
[Fri May 04 12:34:27 2007] 00001660 000019d4 - TRACE: ws_common:
websphereFindTransport: Finding the transport
[Fri May 04 12:34:27 2007] 00001660 000019d4 - DETAIL: ws_common:
websphereFindTransport: Setting the transport(case 2): IBM99TVXRD.au.ibm.com on
port 9081
[Fri May 04 12:34:27 2007] 00001660 000019d4 - TRACE: ws_common:
websphereExecute: Executing the transaction with the app server
...
```

```
[Fri May 04 12:34:27 2007] 00001660 000019d4 - TRACE: ws_common:
websphereExecute: Wrote the request; reading the response
[Fri May 04 12:34:27 2007] 00001660 000019d4 - DETAIL: lib_htresponse:
htresponseRead: Reading the response: 4b1b18c
[Fri May 04 12:34:27 2007] 00001660 00001388 - DEBUG: ws_common:
websphereGetStream: socket 260 connected to IBM99TVXRD.au.ibm.com:9081
[Fri May 04 12:34:27 2007] 00001660 00001388 - DEBUG: lib_stream: openStream:
Opening the stream
```

The last entry for the thread ID (000019d4) is:

```
Fri May 04 12:34:27 2007] 00001660 000019d4 - DETAIL:
lib_htresponse: htresponseRead: Reading the response: 4b1b18c
```

The plug-in log then shows processing for a new request as shown by the change in the thread ID to 00001388. Searching down through the log will not show a response for that particular request and thread ID.

You will need to look into the JVM logs to see why the application server is not responding.

Analyze the JVM logs

Look for the following:

- If you are unable to attach to the host to view the logs, the server has crashed.

  Resolution of this problem is outside the scope of this chapter, but you can tune the plug-in to better recognize this situation. Go to “Host down” on page 249.

- WSVR0605W messages indicating hung threads.

  The following message indicates your application server has hung or is in the process of hanging:

  ```
  [1/05/07 15:19:57:873 EST] 0000001f ThreadMonitor W   WSVR0605W:
  Thread "WebContainer : 1" (0000003f) has been active for 695740
  milliseconds and may be hung.  There is/are 99 thread(s) in total in
  the server that may be hung.
  ```

  You might also see the process still running but no further logging activity.

  Resolution of this problem is outside the scope of this chapter, but you can tune the plug-in to better recognize this situation. Go to “Hung application server” on page 249.

6.3.3 Root causes

The following are possible root causes that would lead to these issues. The resolution of these root causes is outside the scope of this chapter. However you can tune the plug-in to better detect and work around the problem.

- Host down
- Hung application server

6.3.4 Host down

If the server host is completely down, the plug-in can take some time to recognize this. In this case, the plug-in will attempt to connect to the server but since it is no longer responding to TCP requests, the plug-in will need to wait until the operating system times out the connection request before marking the server down. Depending on the operating system, this can take up to 3 minutes.

Resolve the problem

The plug-in can be tuned to better recognize a server host that has gone down by adjusting the `ConnectTimeout` parameter.

Tuning the `ConnectTimeout` parameter is discussed in the technote:

- Understanding HTTP plug-in failover in a clustered environment

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219808

6.3.5 Hung application server

If an application server has hung, it can still be possible for it to accept incoming HTTP requests yet do no work with them. In this case, the plug-in will be able to establish a connection with the application server and so will not recognize that it has failed. It will send the request across and wait on a response. In this instance, attempting to connect to the applications server's HTTP transport port will cause your browser to sit and wait for a response.

Resolve the problem

The plug-in can be tuned to mark a hung server down. Adjust the `ServerIOTimeout` parameters to tune the plug-in so that it recognizes a hung server as an error condition.

`ServerIOTimeout` is discussed in the technote:

- Understanding HTTP plug-in failover in a clustered environment

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219808

6.3.6 Validate the solution

After making the changes described here, recreate the issue and ensure that the plug-in marks the server down in a timely manner. this will enable the cluster members to carry on processing while you resolve the application server problems.

If you are still experiencing an issue with the plug-in not marking the server down, go to "The next step" on page 264.

6.4 Plug-in not recovering from a server outage

This section discusses steps to diagnose the problem where following a server failure, the plug-in marked the server down and stopped routing requests to that server. You have restored the application server but the plug-in has not started sending it requests.

6.4.1 Collect diagnostics

You will need to collect and review a plug-in trace to see why the plug-in is not sending requests to the restored server. You may also need to review the JVM logs from the application server.

Depending on the plug-in logging level you have set, you may need to set it higher and recreate the problem to collect the data.

Collect the following:

- The plug-in log where the logging level has been set to a higher level than Error.

  See 9.3, "Collecting the plug-in log" on page 317.

- JVM SystemOut and SystemErr logs

  See 9.1, "Collecting JVM logs" on page 314.

6.4.2 Analyze diagnostics

You will first need to review the plug-in log. Depending on what you see, you may also need to review the JVM logs.

Analyze the plug-in trace

Look for the following in the trace:

- Setting the plug-in logging level to `Stats` will allow you to see when an application server starts responding to requests, you will first see the server being marked down. While it is down, there will be no stats lines written to the `http-plugin.log` file for that server. When the plug-in starts routing requests to the server, you will again see stats lines for the server.
- Setting the plug-in logging level to `Detail` or higher will show you more detailed information about the process the plug-in is taking to check a server that has been marked down as shown in Example 6-9. First, you can see server1 being marked down. Processing the next request shows that the plug-in will retry `server1` in 5 seconds. Approximately 5 seconds later, the message reports that the server will be retried now. This retry fails. Finally, you can see the last request in the example being successfully routed to `server1`.

Example 6-9 Checking to see if a server is available

```
[Fri Apr 20 13:02:04 2007] 00001e44 00001058 - ERROR: ws_server:
serverSetFailoverStatus: Marking IBM99TVXRDNode1_server1 down
[Fri Apr 20 13:02:04 2007] 00001e44 00001058 - STATS: ws_server:
serverSetFailoverStatus: Server IBM99TVXRDNode1_server1 : pendingRequests 0
failedRequests 1 affinityRequests 0 totalRequests 4.
...
[Fri Apr 20 13:02:09 2007] 00001e44 00001058 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of IBM99TVXRDNode1_server1,
ignoreWeights 0, markedDown 1, retryNow 0, retryInSec 5, wlbAllows 0
reachedMaxConnectionsLimit 0
...
[Fri Apr 20 13:02:14 2007] 00001e44 00001058 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of IBM99TVXRDNode1_server1,
ignoreWeights 0, markedDown 1, retryNow 1, retryInSec 0, wlbAllows 0
reachedMaxConnectionsLimit 0
[Fri Apr 20 13:02:14 2007] 00001e44 00001058 - DETAIL: ws_common:
websphereFindTransport: Setting the transport(case 2): IBM99TVXRD.au.ibm.com on
port 9081
[Fri Apr 20 13:02:14 2007] 00001e44 00001058 - ERROR: ws_common:
websphereGetStream: Failed to connect to app server on host
'IBM99TVXRD.au.ibm.com', OS err=10061
...
```

```
[Fri Apr 20 13:04:04 2007] 00001e44 00001058 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of IBM99TVXRDNode1_server1,
ignoreWeights 0, markedDown 1, retryNow 1, retryInSec 0, wlbAllows 0
reachedMaxConnectionsLimit 0
[Fri Apr 20 13:04:04 2007] 00001e44 00001058 - DETAIL: ws_common:
websphereFindTransport: Setting the transport(case 2): IBM99TVXRD.au.ibm.com on
port 9081
[Fri Apr 20 13:04:04 2007] 00001e44 00001058 - DETAIL: ws_common:
websphereGetStream: Created a new stream; queue was empty, socket = 620
```

If you see the plug-in still not being able to connect to the restored server, you will need to review the JVM logs.

If you determine that the application server started, but the plug-in is taking too long to recognize this, go to "Tune the retry interval" on page 253.

Analyze the JVM logs

Look for the following:

- Ensure the server has finished starting by looking for the message that indicates a server is ready to process requests:

  ```
  [3/05/07 17:01:21:959 EST] 0000000a WsServerImpl  A   WSVR0001I:
  Server ejbserver1 open for e-business
  ```

  If this message has not appeared, either wait for the server to finish startup or try restarting it again.

- Ensure the application has started correctly:

  If the application fails to start, you will see exceptions that will describe what has gone wrong with the application on startup. Example 6-10 shows the snoop servlet failing to start due to a `ClassNotFoundException`.

Example 6-10 Application fails to start

```
[4/05/07 13:16:55:509 EST] 00000023 ServletWrappe E   [Servlet
Error]-[SnoopServlet]: java.lang.ClassNotFoundException: SnoopServlet
   at
com.ibm.ws.classloader.CompoundClassLoader.findClass(CompoundClassLoader.java:4
72)
   at
com.ibm.ws.classloader.CompoundClassLoader.loadClass(CompoundClassLoader.java:3
73)
```

Resolution of this problem is outside the scope of this paper, but you can tune the plug-in timing of when to check if a server has become available.

6.4.3 Root causes

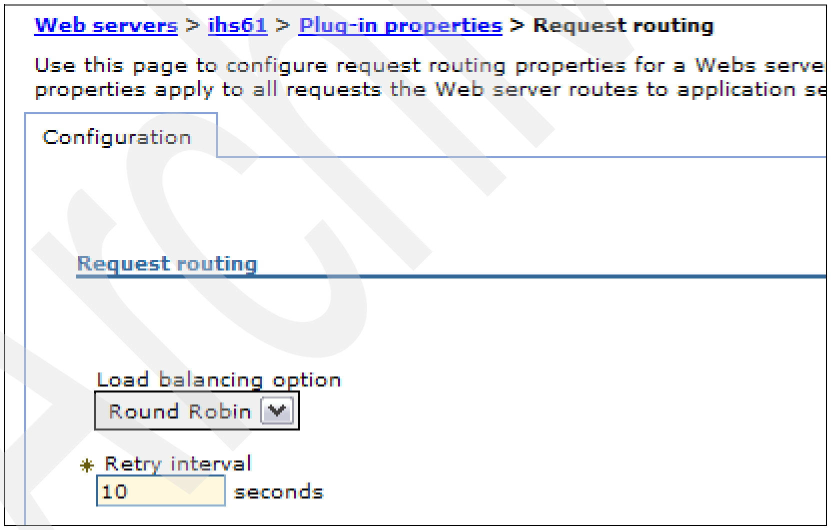
The root cause of this issue is not a problem with plug-in load balancing, however you can tune the plug-in to respond to a restored server in a more timely manner by tuning the retry interval.

6.4.4 Tune the retry interval

When an application server comes back online, the plug-in will automatically recognize this and start to route requests back to that server. The plug-in does not log a message to indicate a server is back online, it simply starts routing requests to that server.

The timing of how often the server checks to see if a server has become available is controlled by the `Retry interval` parameter set in the administrative console as shown in Figure 6-5. **Navigate to Servers → Web servers → *server_name* → Plug-in properties → Request routing**.

You can also see the selected value in the cluster definition in the `plugin-cfg.xml` file.


Web servers > ihs61 > Plug-in properties > Request routing

Use this page to configure request routing properties for a Webs serve
properties apply to all requests the Web server routes to application se

Load balancing option: Round Robin

* Retry interval: 10 seconds

Figure 6-5 Retry interval

Resolve the problem

Tune the retry interval to suit your environment. If the servers typically take a minute to start, set the retry interval to say 30 seconds. If you set the interval too high, you might have an idle server while the plug-in waits for the retry interval to elapse. If you set the retry interval too low, the plug-in will check too often to see if a server has become available.

6.4.5 Validate the solution

After tuning the plug-in retry interval, recreate the failure and restore the server. Ensure that the plug-in responds to a restored server in a timely manner. Also, ensure that you are not checking for a restored server too often as this could be cause a performance impact.

If you still see problems with the server not recognizing a restored server, go to “The next step” on page 264.

6.5 Session affinity not working

This section discusses steps to diagnose your problem if your application relies on session affinity to maintain user state information, but you are not seeing session affinity being honoured.

6.5.1 Collect diagnostics

You will need to collect and review a plug-in trace to see why the plug-in is not sending requests to the correct servers. You may also need to review the JVM logs from the application server.

Depending on the plug-in logging level you have set, you may need to set it higher and recreate the problem to collect the data.

Collect the following:

- The plug-in log where you have set the logging level to Trace.

  See 9.3, “Collecting the plug-in log” on page 317.
- JVM SystemOut and SystemErr logs

  See 9.1, “Collecting JVM logs” on page 314.

6.5.2 Analyze diagnostics

You will first need to review the plug-in log. Depending on what you see, you may also need to review the JVM logs.

Analyze the plug-in trace

Refer to “Analyze the plug-in detail logging” on page 237 for details on plug-in tracing.

Look for the following:

- Detailed information about the processing of session affinity requests.

  Review the trace to determine why a particular request was sent to a particular server. Example 6-11 shows an example of a request to the HitCount servlet being routed to **server1** based on the partition ID located in the **JSESSIONID** cookie.

Example 6-11 Processing the HitCount servlet

```
[Tue Apr 17 16:49:57 2007] 00001b74 00000698 - DETAIL: ws_common:
websphereShouldHandleRequest: trying to match a route for: vhost='localhost';
uri='/hitcount'
...
[Tue Apr 17 16:49:57 2007] 00001b74 00000698 - DEBUG: ws_common:
websphereParseCloneID: Parsing clone ids from
'0001WxG_w55xP_kcCK-5-PqZt3A:34J7RVRCR5'
[Tue Apr 17 16:49:57 2007] 00001b74 00000698 - DEBUG: ws_common:
websphereParseCloneID: Parsing clone ids from
'0001WxG_w55xP_kcCK-5-PqZt3A:34J7RVRCR5'
[Tue Apr 17 16:49:57 2007] 00001b74 00000698 - DEBUG: ws_server_group:
serverGroupMatchPartitionID: Match found for partitionID '34J7RVRCR5'
[Tue Apr 17 16:49:57 2007] 00001b74 00000698 - DEBUG: ws_server_group:
serverGroupGetServerByID: Match for clone 'IBM99TVXRDNode1_server1'
```

  The trace will also show you how the partition table is constructed as shown in Example 6-12. The partition table is populated when the first request is processed by the plug-in. It communicates with the application servers and builds the partition table based on the session failover configuration. The plug-in gets the partition table from the application servers as it needs to be able to maintain a consistent partition table to handle session failover among multiple HTTP servers or following an HTTP server restart.

Example 6-12 Plug-in building the partition table

```
[Tue Apr 17 16:57:30 2007] 00001584 000014b8 - TRACE: ws_common:
ParsePartitionIDs: Parsing partitionID pair from
'8I9FV6LVH:1251sllkq;34J7RVRCR5:12522k3ef;'
[Tue Apr 17 16:57:30 2007] 00001584 000014b8 - TRACE: ws_common:
ParsePartitionIDs: Adding partitionID / clone pair '8I9FV6LVH' : '1251sllkq'
[Tue Apr 17 16:57:30 2007] 00001584 000014b8 - TRACE: ws_common:
ParsePartitionIDs: Adding partitionID / clone pair '34J7RVRCR5' : '12522k3ef'
```

```
[Tue Apr 17 16:57:30 2007] 00001584 000014b8 - TRACE: ws_common:
ParsePartitionIDs: Returning partitionID / cloneid pair list
```

Being able to see how the partition table is setup will allow you to cross reference the partition ID to the clone ID of each application server. If you are not seeing a partition table, you should be seeing session affinity being maintained by the clone ID.

If you are not seeing the session ID being passed back from the client, for example, in the `JSESSIONID` cookie, go to "Session ID missing" on page 256.

If you are seeing that the session ID can not be matched to an existing clone ID or partition ID, go to "Invalid clone ID" on page 257.

If you are not seeing the cause of the problem in the plug-in trace, then you will need to review the JVM logs.

Analyze the JVM logs

Look for the following:

- Ensure that the application servers are processing the sessions correctly. You will need to take an HTTP session trace in WebSphere Application Server.

Debugging session handling problems in WebSphere Application Server applications is outside the scope of this document.

6.5.3 Root causes

The following are possible root causes for this issue:

- Session ID missing
- Invalid clone ID

6.5.4 Session ID missing

When an HTTP session is established, that session is assigned an ID and the ID is passed to the client. In subsequent requests, the session ID is passed back from the client so that the application can keep track of the session.

Resolve the problem

Ensure that the session ID can be passed back from the client using one of the available methods. WebSphere Application Server can maintain session information using any or all of the following:

- Cookies

  The session ID is encoded in a session cookie called **JSESSIONID** by default.The client browsers must be set to accept cookies for this to work.

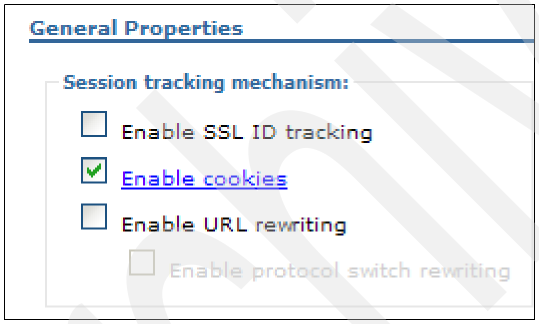
- URL rewriting

  If cookies are not available, sessions can be maintained by rewriting the URL to include the session ID as an HTTP parameter. This introduces a performance impact.

- SSL ID tracking

  The session identifier is linked to SSL identifier when SSL is used to secure browser to application server communications.

The allowed session tracking mechanisms are configured for the application servers in the administrative console. Navigate to **Servers** → **Application servers** → ***server_name*** → **Session management**. Figure 6-6 shows cookies set as the session tracking mechanism.


General Properties

Session tracking mechanism:

- [ ] Enable SSL ID tracking
- [✓] Enable cookies
- [ ] Enable URL rewriting
  - [ ] Enable protocol switch rewriting

Figure 6-6 Session tracking mechanisms

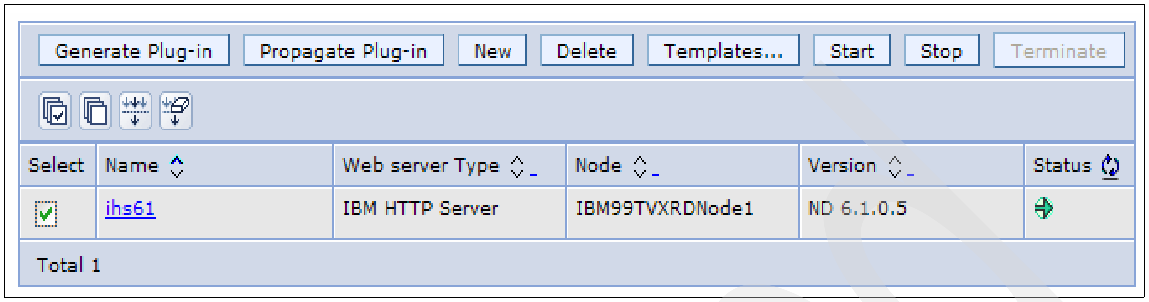
6.5.5 Invalid clone ID

If the session ID being returned from the client contains a clone ID that does not match the clone IDs configured in the plug-in configuration file, then session affinity will not be maintained.

Resolve the problem

Ensure the session IDs and clone IDs being reported in the plug-in trace are valid. If the clone IDs have changed or the plug-in has been manually edited, then these values may no longer be valid. Regenerate and distribute the plug-in to ensure it reflects the latest configuration.

In the administrative console. navigate to **Servers** → **Web servers**. Select the Web server name as shown in Figure 6-7 and click **Generate Plug-in**. Click the server name again and click **Propagate plug-in**.



Select	Name	Web server Type	Node	Version	Status
✓	ihs61	IBM HTTP Server	IBM99TVXRDNode1	ND 6.1.0.5	●

Total 1

Figure 6-7 Generate the plug-in configuration file

6.5.6 Validate the solution

Perform the steps described in this section and recreate the problem. Ensure that your sessions are being maintained across requests as you expect.

If you are still experiencing problems with session affinity, go to “The next step” on page 264.

6.6 Sessions not failing over under error conditions

WebSphere Application Server allows you to configure your system so that session data is shared between the application servers in the cluster. If one of the servers fails, then a subsequent request for a session that was being handled by the failed application server can be picked up by another application server and processing continue.

There are two recommended mechanisms for handling the sharing of session data:

- The first and recommended mechanism is memory-to-memory replication. You create a session replication domain in your cluster and allow sessions to be replicated between the application servers.
- The second method is to maintain session data in database that is accessible to all servers in a cluster.

6.6.1 Collect diagnostics

To determine why sessions are not failing over, you will need to set the plug-in logging level to Trace, recreate the problem and collect the http-plugin.log and JVM logs.

Collect the following

- The plug-in log where you have set the logging level to Trace.

  See 9.3, "Collecting the plug-in log" on page 317.
- JVM SystemOut and SystemErr logs

  See 9.1, "Collecting JVM logs" on page 314.

6.6.2 Analyze diagnostics

You will first need to review the plug-in log. Depending on what you see, you may also need to review the JVM logs.

Plug-in trace

Refer to "Analyze the plug-in detail logging" on page 237 for details on plug-in tracing.

Look for the following:

- Review a plug-in trace to track a request that you expect to be failing over to another application server.

  Example 6-13 shows the plug-in extracting the partition ID from the session cookie and finding the server that should process this request; that is, `server1`.

Example 6-13 Session failover, selecting the server

```
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the partitionID in cookie JSESSIONID:
0001WxG_w55xP_kcCK-5-PqZt3A:34J7RVRCR5
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - DEBUG: ws_common:
websphereParseCloneID: Parsing clone ids from
'0001WxG_w55xP_kcCK-5-PqZt3A:34J7RVRCR5'
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_common:
websphereParseCloneID: Adding clone id '34J7RVRCR5'
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_common:
websphereParseCloneID: Returning list of clone ids
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupFindDwlmServer: Looking for dwlm pair
```

```
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupMatchPartitionID: Looking for partitionID
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupMatchPartitionID: Comparing curCloneID '34J7RVRCR5' to partitionID
'8I9FV6LVH'
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupMatchPartitionID: Comparing curCloneID '34J7RVRCR5' to partitionID
'34J7RVRCR5'
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - DEBUG: ws_server_group:
serverGroupMatchPartitionID: Match found for partitionID '34J7RVRCR5'
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupGetFirstServer: getting the first server
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupGetServerByID: Comparing curCloneID '12522k3ef' to server clone id
'12522k3ef'
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - DEBUG: ws_server_group:
serverGroupGetServerByID: Match for clone 'IBM99TVXRDNode1_server1'
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupFindDwlmServer: Match for clone 'IBM99TVXRDNode1_server1'
primary server
```

Example 6-14 shows the plug-in trying to connect to `server1` to maintain session affinity. The attempt fails because `server1` is down. In the message, `OS error 10061` refers to the Windows TCP connection refused error.

Example 6-14 Failed connection

```
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_server_group:
lockedServerGroupUseServer: Server IBM99TVXRDNode1_server1 picked, weight 0.
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_common:
websphereFindTransport: Finding the transport
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - DETAIL: ws_common:
websphereFindTransport: Setting the transport(case 2): IBM99TVXRD.au.ibm.com on
port 9081
[Tue Apr 17 17:06:30 2007] 00001584 000014b8 - TRACE: ws_common:
websphereExecute: Executing the transaction with the app server
...
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - ERROR: ws_common:
websphereGetStream: Failed to connect to app server on host
'IBM99TVXRD.au.ibm.com', OS err=10061
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_common:
websphereGetStream: socket 644 closed - failed to connect
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - ERROR: ws_common:
websphereExecute: Failed to create the stream
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - ERROR: ws_server:
serverSetFailoverStatus: Marking IBM99TVXRDNode1_server1 down
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - STATS: ws_server:
serverSetFailoverStatus: Server IBM99TVXRDNode1_server1 : pendingRequests 0
failedRequests 1 affinityRequests 1 totalRequests 1.
```

```
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - ERROR: ws_common:
websphereHandleRequest: Failed to execute the transaction to
'IBM99TVXRDNode1_server1'on host 'IBM99TVXRD.au.ibm.com'; will try another one
```

Finally, the plug-in goes back to the session cookie and checks for another available server to handle the request and resume the user session based on the partition ID as shown in Example 6-15. Note also the plug-in retrieving an updated partition table from the application server.

Example 6-15 Session failover

```
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_common:
websphereHandleSessionAffinity: Look for partitionID in affinity cookie
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the partitionID in cookie JSESSIONID:
0001WxG_w55xP_kcCK-5-PqZt3A:34J7RVRCR5
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - DEBUG: ws_common:
websphereParseCloneID: Parsing clone ids from
'0001WxG_w55xP_kcCK-5-PqZt3A:34J7RVRCR5'
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_common:
websphereParseCloneID: Adding clone id '34J7RVRCR5'
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_common:
websphereParseCloneID: Returning list of clone ids
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupFindDwlmServer: Looking for dwlm pair
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - DEBUG: ws_server_group:
serverGroupNextRoundRobinServer: Round Robin load balancing
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupNextRoundRobinServer: numPrimaryServers is 2
...
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_server_group:
lockedServerGroupUseServer: Server IBM99TVXRDNode2_server2 picked,
weight 0.
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupIncrementConnectionCount: Server IBM99TVXRDNode2_server2 picked,
pendingConnectionCount 1 totalConnectionsCount 1.
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - DEBUG: ws_server_group:
serverGroupNextRoundRobinServer: use server IBM99TVXRDNode2_server2
[Tue Apr 17 17:06:31 2007] 00001584 000014b8 - TRACE: ws_server_group:
serverGroupFindDwlmServer: Retrieve updated dwlm partition table from server
IBM99TVXRDNode2_server2 (dwlmStatus = 22)
```

If you are not seeing a partition table, session replication may not be correctly configured. Go to “Session replication incorrectly configured” on page 262.

If you are not seeing the cause of the problem in the plug-in trace, then you will need to review the JVM logs.

Analyze the JVM logs

Look for the following:

- Ensure that the application servers are processing the sessions correctly. You will need to take an HTTP session trace in WebSphere Application Server.

Debugging session handling problems in WebSphere Application Server applications is outside the scope of this document.

6.6.3 Root causes

The following is a possible root cause for this issue:

- Session replication incorrectly configured

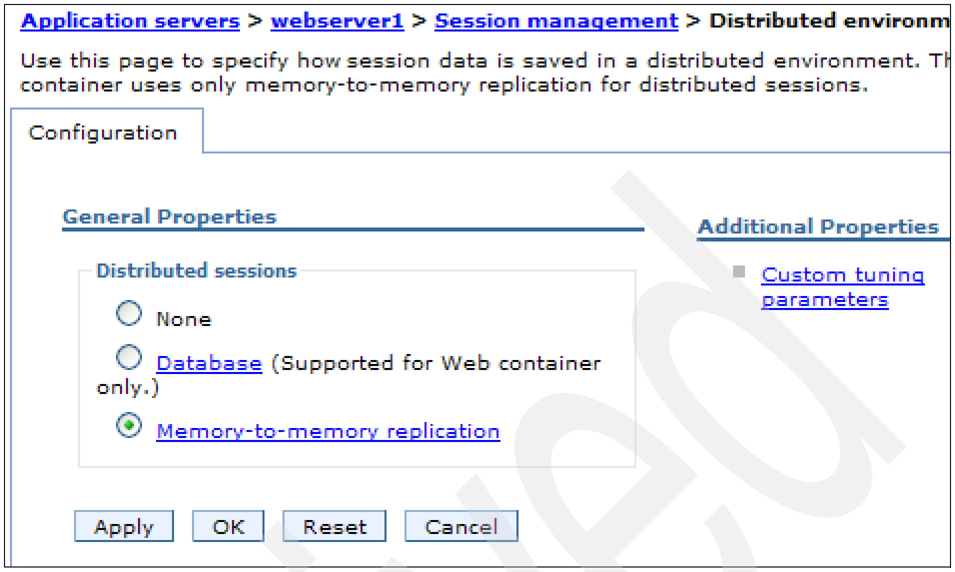
6.6.4 Session replication incorrectly configured

For the plug-in to be able to failover sessions to another application server, you need to configure session replication in the administrative console for each application server in the cluster.

Resolve the problem

Ensure there is a valid session replication domain defined and that all cluster members are configured to use this replication domain. If there is no valid session replication domain, there will be no partition table details reported in the plug-in log.

Navigate to **`Servers`** → **`Application servers`** → ***server_name*** → **`Session management`** → **`Distributed environment settings`**. Figure 6-8 shows that memory to memory session replication is enabled for the application server.


Application servers > webserver1 > Session management > Distributed environm

Use this page to specify how session data is saved in a distributed environment. Th
container uses only memory-to-memory replication for distributed sessions.

Configuration

General Properties

Distributed sessions

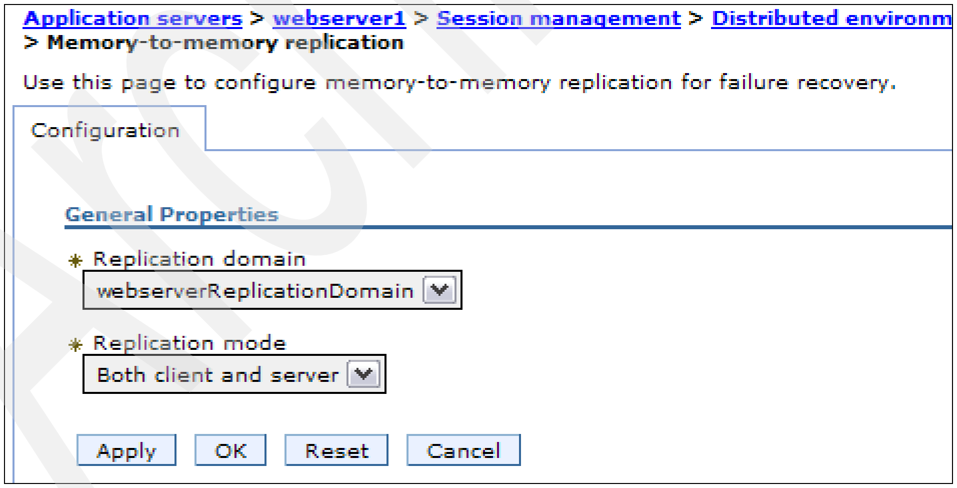
None

Database (Supported for Web container only.)

[✓ Memory-to-memory replication]

Figure 6-8 Memory to memory session replication enabled

Figure 6-9 shows the name of the session replication domain and the replication mode. From the `Distributed environment settings` page in the admin console, click **`Memory-to-memory replication`** to review this setting.


Application servers > webserver1 > Session management > Distributed environm
> Memory-to-memory replication

Use this page to configure memory-to-memory replication for failure recovery.

Configuration

General Properties

* Replication domain: webserverReplicationDomain

* Replication mode: Both client and server

Figure 6-9 Memory to memory session replication settings

All servers that are to participate in a session replication domain must be configured with the same `Replication domain` name.

6.6.5 Validate the solution

Ensure your system is correctly configured to enable session replication and recreate the issue. You will need to restart the application servers for any changes to take effect.

If you are still experiencing an issue, go to “The next step” on page 264.

6.7 The next step

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, there are other things that can go wrong with plug-in load balancing.

6.7.1 Java heap problems

Guidance on Java heap problems can be found in the following references:

- The IBM Guided Activity Assistant tech preview for the IBM Support Assistant has a Java memory troubleshooting activity. For more information about downloading and installing this diagnostic tool, see:

  `http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24011818`

- *Diagnosing out-of-memory errors and Java heap memory leaks*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/ttrb_mdd4j.html`

- *Memory leak detection and analysis in WebSphere Application Server: Part 2: Tools and features for leak detection and analysis*

  `http://www-128.ibm.com/developerworks/websphere/library/techarticles/0608_poddar/0608_poddar.html`

6.7.2 Search online support

If you are sure the problem is with the plug-in, there are things that you can do before contacting IBM support. First, you should review the documentation that you have gathered for errors that were not addressed in this paper and search support sites for information or fixes.

Look for current information available from IBM support on known issues and resolutions using the following search argument:

http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCC2GP&q=MustGather

Look also at the WebSphere Information Center documentation and other technotes for additional resources for diagnosing and fixing plug-in work load management issues:

- *Communicating with Web servers*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/twsv_plugin.html

- *Web server plug-in configuration properties*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/rwsv_plugin_propstable.html

- *Understanding IBM HTTP Server plug-in load balancing in a clustered environment*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219567

- *WebSphere plugin failover delayed when system is physically unavailable*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21052862

- *Understanding HTTP plug-in failover in a clustered environment*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219808

6.7.3 Re-evaluate the symptoms

If, after going through this process, you still have an undiagnosed problem, we recommend that you go back to *Approach to Problem Determination in WebSphere Application Server V6* at:

http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf

Review the problem classifications to see if there are any other components that might be causing the problem.

6.7.4 Contact IBM

There is currently no MustGather documentation for plug-in problems in WebSphere Application Server V6.1. If these steps do not resolve your problem, contact IBM technical support for assistance.

EJB workload management problem determination

The EJB Workload Management (WLM) component in WebSphere Application Server provides routing services for incoming application requests so that these can be distributed to Enterprise JavaBeans™ (EJBs). WLM also provides failover capabilities among the servers hosting the EJB. An EJB client is a process that makes a call to an EJB container to process an EJB request. The following types of EJB clients participate in EJB WLM automatically:

- Clients in the same application server (servlets, JSPs, EJBs)
- Clients in a different application server (servlets, JSPs, EJBs)
- Java applications running within a WebSphere client container
- Stand-alone Java applications using the WebSphere-supplied Java Runtime Environment (JRE™)

EJB WLM uses a weighted round robin algorithm to decide which server to send a request to. When you create a cluster, all cluster members are assigned a weight of 2 so that requests should be evenly distributed among the cluster members. You have the option of modifying the weight so that powerful servers are sent more requests for processing than less powerful servers.

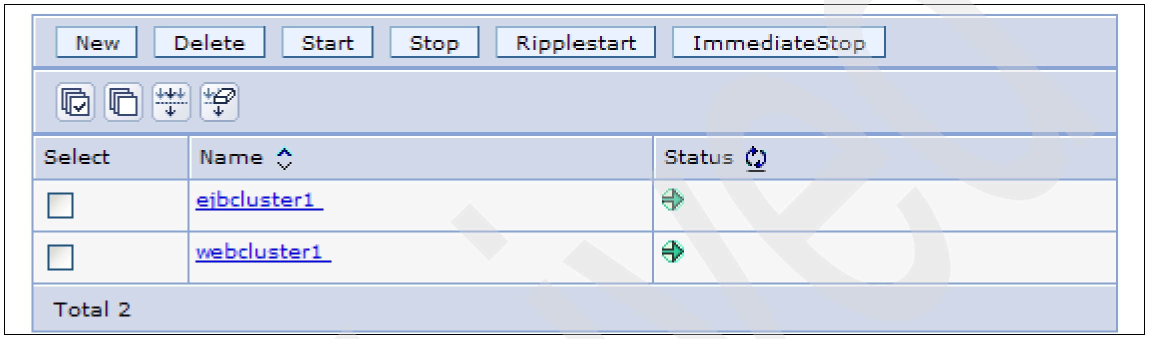
Problems with the EJB WLM can cause requests to fail or requests to be incorrectly routed. This can lead to performance degradation.

7.1 Check system integrity

If you think you might have a EJB WLM problem, the first step is to verify the system integrity:

- Verify that all cluster member servers are up and running.

  In the administrative console, navigate to **`Servers`** → **`Clusters`** and ensure your clusters are started (as shown in Figure 7-1). All cluster members need to be started to ensure even load balancing.



Select	Name	Status
☐	ejbcluster1	●
☐	webcluster1	●

Total 2

Figure 7-1 Started clusters

- Ensure that all the machines in your configuration have TCP/IP connectivity to each other by running the **`ping`** or **`telnet`** command:
  - From each physical server to the deployment manager
  - From the deployment manager to each physical server
  - From the client to each physical server
  - Between all physical servers
- If possible, try accessing the enterprise bean directly on the problem server to see if there is a problem with TCP/IP connectivity, application server health, or other problem not related to workload management. You will need a custom EJB client to achieve this.

7.2 Identify symptoms

The following are typical symptoms of an EJB WLM problem.

- EJB application requests do not get serviced
- EJB requests are not distributed evenly or to all servers

- Failing server still receives EJB requests (failover fails)
- Restarted servers do not share the workload

7.3 EJB application requests do not get serviced

This section discusses steps to diagnose your problem if your EJB client is making a request to an EJB that is running in a clustered environment, but that request is not getting serviced. Another symptom is that you are getting CORBA runtime exceptions returned to the client.

7.3.1 Collect diagnostics

When load balancing requests across a cluster, you do not know in advance which server will handle a given request. For this reason. you will need to review the logs from all the application servers in the cluster along with logs from the EJB client making the call.

Depending on the problem you are investigating, you should consider simplifying the problem determination process by reducing the number of servers to monitor while debugging the problem. You would do this by shutting down all but the minimum number of servers required to recreate the issue.

The following diagnostic data is useful in diagnosing the problem.

- JVM SystemOut and SystemErr logs for both the EJB client process and the EJB server processes.

  See 9.1, "Collecting JVM logs" on page 314.
- First Failure Data Capture (FFDC) incident logs

  The FFDC logs often provide a greater level of detail about a given exception and can be quite useful in debugging EJB failures. You should see a message written in the SystemOut.log pointing you to the specific FFDC log if one has been generated for your problem.
- Activity.log
- Network trace

You can collect it all at once, or start by collecting the JVM logs and then determining whether one or more of the other logs and traces are needed.

7.3.2 Analyze diagnostics

You first need to look at the JVM logs for both the EJB client process and the EJB server processes. These logs may not immediately show you the cause of the error but may refer you to further messages in the FFDC logs. The activity log may also give more information that will help you debug the problem.

If you cannot find the source of the problem from the logs, you may need to track the EJB request from the client to the server using a network analyzer to see if the problem is communications related.

Analyze the JVM logs

Look for the following:

- `org.omg.CORBA.NO_IMPLEMENT: No cluster data` errors on first access to EJB in a cluster.

  If you see this error, go to "No cluster data" on page 273.
- `org.omg.CORBA.*` errors.

  If you see this error, go to "CORBA errors" on page 274.
- You may also find that a problem accessing an EJB cluster does not get reported as such in the JVM logs. You may find messages indicating other problems that end up being an EJB WLM problem. For example: The messages shown in Example 7-1 may appear in the application server logs.

Example 7-1 JVM error messages

```
[1/05/07 15:14:22:391 EST] 0000003f ServiceLogger I
com.ibm.ws.ffdc.IncidentStreamImpl initialize FFDC0009I: FFDC opened
incident stream file
c:\IBM\WAS61\AppServer\profiles\Node1\logs\ffdc\webserver1_66126612_07.
05.01_15.14.22_0.txt
[1/05/07 15:14:23:052 EST] 0000003f ServiceLogger I
com.ibm.ws.ffdc.IncidentStreamImpl resetIncidentStream FFDC0010I: FFDC
closed incident stream file
c:\IBM\WAS61\AppServer\profiles\Node1\logs\ffdc\webserver1_66126612_07.
05.01_15.14.22_0.txt
[1/05/07 15:19:57:873 EST] 0000001f ThreadMonitor W   WSVR0605W: Thread
"WebContainer : 1" (0000003f) has been active for 695740 milliseconds
and may be hung.  There is/are 1 thread(s) in total in the server that
may be hung.
```

Note that the `WSVR0605W` message appearing in this example is usually not associated with an EJB WLM problem but with a hung thread issue in the application server. However, the reference to the FFDC0009I and `FFDC0010I` messages tell us that a FFDC incident has been logged. Refer to “Analyze the FFDC incident log” to see how reviewing this incident file identified the problem as EJB WLM related.

Analyze the FFDC incident log

The application server logs should alert you to the existence of a particular FFDC log that would contain further details of the problem. From Example 7-1 above, you can see the reference to the FFDC logs. Opening this log shows you the root cause of the problem as being no cluster members were available to service the request. This is shown in Example 7-2.

Example 7-2 Excerpt from the FFDC log

```
------Start of DE processing------ = [1/05/07 15:14:22:371 EST] , key =
com.ibm.ws.cluster.router.selection.NoClusterMembersAvailableException
com.ibm.ws.cluster.router.selection.SelectionManager.getTarget 254
Exception =
com.ibm.ws.cluster.router.selection.NoClusterMembersAvailableException
Source = com.ibm.ws.cluster.router.selection.SelectionManager.getTarget
probeid = 254
Stack Dump =
com.ibm.ws.cluster.router.selection.NoClusterMembersAvailableException
```

To resolve this problem, go to “CORBA errors” on page 274.

There are other possible causes for this type of problem and the FFDC logs are a useful source of debugging information. If you are unable to determine the root cause of the problem from the FFDC incident log, review the activity log for errors reported around the time of the problem.

Analyze the activity log

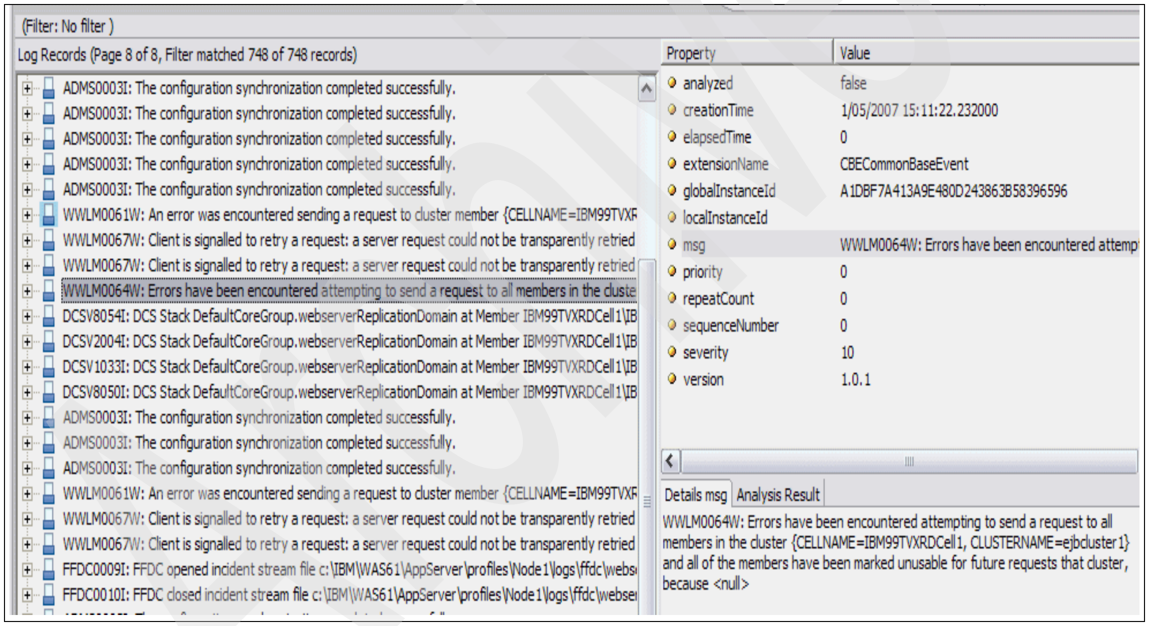
The activity log is a binary log and should be reviewed using the Profiling and Logging perspective in either the Application Server Toolkit or Rational Application Developer.

To view the activity log in the Application Server Toolkit:

1. Start the tool from the Windows command prompt or from the start script.
2. From the **Window** menu, choose **Open Perspective** → **Other** and select **Profiling and Logging**.
3. From the **File** menu choose **Import**. This brings up the `Import` dialog.
4. Select **Log file** and click **Next**.
5. On the next page choose **Add**, this brings up the `Add Log File` dialog.

6. In the `Selected log files` box, choose **`IBM WebSphere Application Server activity log`**.
7. Click the **`Details`** tab and enter the following data:
   - `IBM WebSphere Application Server activity log file`: path to the activity log
   - `IBM WebSphere Application Server installation directory`: path to the WebSphere Application Server installation
   - `IBM WebSphere Application Server version`: 6.x (rules)
8. Click **`OK`** and then click **`Finish`**.

The activity log can give you detailed information about EJB WLM processing. Look for errors around the time of the problem similar to those shown in Example 7-2. In this example, all cluster members are down and therefore the EJB request cannot be processed.


(Filter: No filter)

Log Records (Page 8 of 8, Filter matched 748 of 748 records)

```
ADMS0003I: The configuration synchronization completed successfully.
ADMS0003I: The configuration synchronization completed successfully.
ADMS0003I: The configuration synchronization completed successfully.
ADMS0003I: The configuration synchronization completed successfully.
ADMS0003I: The configuration synchronization completed successfully.
WWLM0061W: An error was encountered sending a request to cluster member {CELLNAME=IBM99TVXR
WWLM0067W: Client is signalled to retry a request: a server request could not be transparently retried
WWLM0067W: Client is signalled to retry a request: a server request could not be transparently retried
WWLM0064W: Errors have been encountered attempting to send a request to all members in the custe
DCSV8054I: DCS Stack DefaultCoreGroup.webserverReplicationDomain at Member IBM99TVXRDCell1\IB
DCSV2004I: DCS Stack DefaultCoreGroup.webserverReplicationDomain at Member IBM99TVXRDCell1\IB
DCSV1033I: DCS Stack DefaultCoreGroup.webserverReplicationDomain at Member IBM99TVXRDCell1\IB
DCSV8050I: DCS Stack DefaultCoreGroup.webserverReplicationDomain at Member IBM99TVXRDCell1\IB
ADMS0003I: The configuration synchronization completed successfully.
ADMS0003I: The configuration synchronization completed successfully.
ADMS0003I: The configuration synchronization completed successfully.
WWLM0061W: An error was encountered sending a request to cluster member {CELLNAME=IBM99TVXR
WWLM0067W: Client is signalled to retry a request: a server request could not be transparently retried
WWLM0067W: Client is signalled to retry a request: a server request could not be transparently retried
FFDC0009I: FFDC opened incident stream file c:\IBM\WAS61\AppServer\profiles\Node1\logs\ffdc\webs
FFDC0010I: FFDC closed incident stream file c:\IBM\WAS61\AppServer\profiles\Node1\logs\ffdc\webse
```

Property	Value
analyzed	false
creationTime	1/05/2007 15:11:22.232000
elapsedTime	0
extensionName	CBECommonBaseEvent
globalInstanceId	A1DBF7A413A9E480D243863B58396596
localInstanceId	
msg	WWLM0064W: Errors have been encountered attemp
priority	0
repeatCount	0
sequenceNumber	0
severity	10
version	1.0.1

```
WWLM0064W: Errors have been encountered attempting to send a request to all
members in the cluster {CELLNAME=IBM99TVXRDCell1, CLUSTERNAME=ejbcluster1}
and all of the members have been marked unusable for future requests that cluster,
because <null>
```

Figure 7-2 Viewing the activity log with the AST

If you can not find any indication of the problem in the activity log, take a network trace to track the EJB request and reply through the network.

Analyze the network trace

Use a tool appropriate to your operating system to collect a network trace. Some tools include `snoop`, `tcptrace` and `ethereal`. Refer to the following technote for further information about network tracing:

- *How to use packet trace tools iptrace, snoop, tcpdump, ethereal, and nettl*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21175744

Recreate the problem and look for the packets that make up the EJB request. Follow the path of the request from the client to the server and follow the response back. This should show you where the request goes missing if the problem is related to network issues. Also look for the following:

- Network saturation
- Dropped or incorrectly routed packets

If you still can not resolve the problem, refer to "The next step" on page 287.

7.3.3 Root causes

The following are possible root causes for this problem:

- No cluster data
- CORBA errors

7.3.4 No cluster data

The WLM component relies on a routing table that shows which servers can handle the EJB requests. The WLM builds the routing table when the first request for an EJB is received. Depending on network topology, building this table can take some time as the Workload Manager needs to contact every server in the cluster to determine its status. Depending on the configuration, the first or first few requests can fail.

You will see the following error in the node agent logs:

```
org.omg.CORBA.NO_IMPLEMENT: No Cluster Data
```

The problem goes away when the routing table is built and normal processing continues.

Resolve the problem

Refer to the following technote for the resolution to this issue:

- *PK20304: NO_IMPLEMENT ON FIRST REQUEST TO A CLUSTER*

  `http://www.ibm.com/support/docview.wss?rs=180&uid=swg1PK20304`

7.3.5 CORBA errors

Look for the following CORBA messages:

org.omg.CORBA.TRANSIENT: SIGNAL_RETRY

This is a transient exception caused when the workload management routing service tries to route a request to a target server. This is the exception thrown to the client when there is no reply to the request.

Another EJB target may be available to service the request, but the request could not be failed over transparently by the WLM because the completion status was not determined to be "no". In this case the client application needs to determine if they want to resend the request. For example, if the EJB request updates a database table, the WLM is not able to determine if the update occurred or not as it did not get a response. Therefore this is left to the application code to determine.

Resolve the problem

You will need to determine if the application server executing the EJB is still operational. If it is not, restart the application server. If the application server is still operational then it is likely experiencing some other issue such as an application server hang or other similar error and is not a WLM issue.

Either way, the application will need to determine how to recover the transaction.

org.omg.CORBA.NO_IMPLEMENT

This exception is thrown when none of the servers participating in the workload management cluster are available. The routing service cannot locate a suitable target for the request.

For example, if the cluster is stopped or if the application does not have a path to any of the cluster members, then this exception will be thrown back to the client. There are many kinds of NO_IMPLEMENT which can be distinguished by the associated message or minor code with the NO_IMPLEMENT exception:

NO_IMPLEMENT: Retry Limit Reached or Forward Limit Reached

NO_IMPLEMENT: Retry Limit Reached and NO_IMPLEMENT: Forward Limit Reached exceptions are thrown when WLM is attempting to route a request to a server and receives an exception which is considered retryable, or the request is being forwarded to another server. In order to avoid an infinite selection loop, these exceptions will be thrown if errors are received or forwarding is done on the same server ten consecutive times.

Resolve the problem

This is likely to be an application error and should be referred to the application team for resolution. You might be able to get more detail by running a WLM trace (see “Enabling the WLM trace” on page 314).

NO_IMPLEMENT: No Available Target

This is a general exception that means the WLM may have some cluster data but perhaps not all. However with the data currently available, the WLM cannot find a valid target for the request. It is possible that cluster members have been marked unusable or just that it does not have the current data necessary to route the request to server.

Resolve the problem

Validate that all the cluster members are available by checking their status in the administrative console. Start the servers if necessary.

If possible, attempt to connect to the EJB directly on each cluster member and review the application server logs for errors that will indicate the cause of the problem.

NoAvailableTargetException

This exception is internal to IBM only, you may see it printed out in traces with the WLM trace specification enabled, but this exception is internal to the WLM code. This exception is often expected, especially in fail over and startup scenarios and if a real problem exists, it would manifest itself as one of the NO_IMPLEMENT exceptions above.

Resolve the problem

Refer this problem to IBM support as outlined below in “The next step” on page 287.

If an application server is still failing to serve requests and you are not seeing any errors, try to restart the server.

7.3.6 Validate the solution

Recreate the situation and validate that requests are being serviced. Your requests should get serviced and no CORBA related errors should appear in the logs.

If this does not resolve the issue, go to “The next step” on page 287.

7.4 EJB requests are not distributed evenly or to all servers

This section discusses steps to diagnose your problem if you are seeing EJB requests being processed, but not all cluster members appear to be processing the requests.

7.4.1 Collect diagnostics

When load balancing requests across a cluster, you do not know in advance which server will handle a given request. For this reason. you will need to review the logs from all the application servers in the cluster along with logs from the EJB client making the call.

Depending on the problem you are investigating, you should consider simplifying the problem determination process by reducing the number of servers to monitor while debugging the problem. You would do this by shutting down all but the minimum number of servers required to recreate the issue.

Collect the following diagnostic data:

- Performance Monitoring Infrastructure (PMI) data
- JVM SystemOut and SystemErr logs

  See 9.1, “Collecting JVM logs” on page 314.

7.4.2 Analyze diagnostics

First determine the distribution of EJB requests among your cluster members by analyzing the EJB WLM statistics available from the PMI. If you identify a server not processing any requests or uneven load balancing, review the JVM logs to see if there are any errors or problems. You should also review server resource usage to determine if the problem is caused by server load.

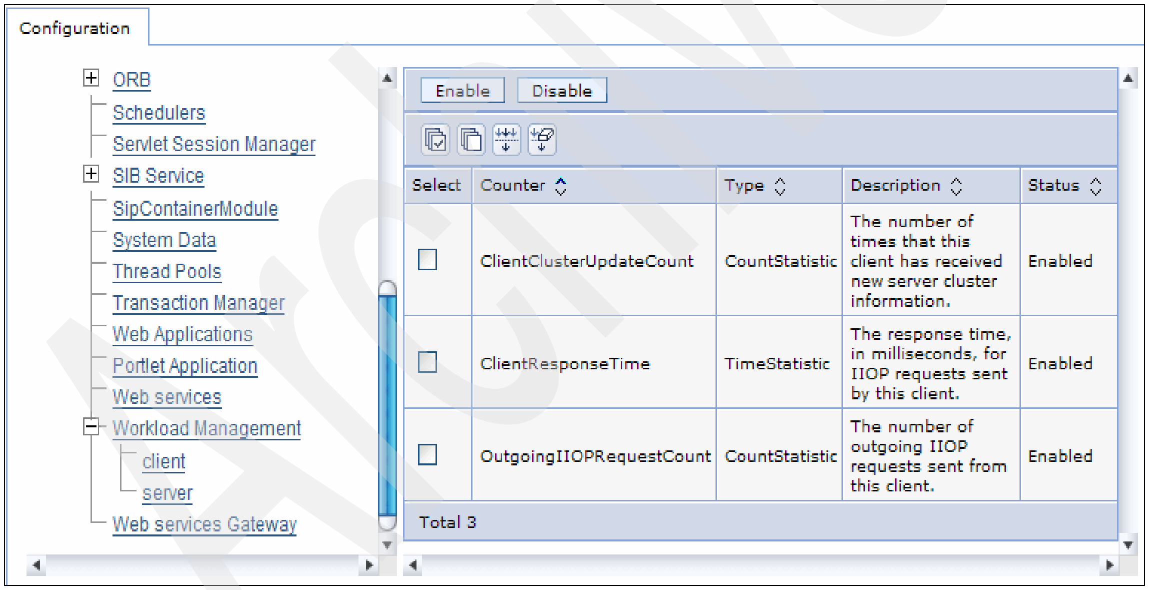
Analyze the PMI data

You first need to ensure that the server is set to collect these PMI statistics.

For each application server you want to monitor:

1. Navigate to **Monitoring and tuning** → **Performance Monitoring Infrastructure (PMI)** and choose your server name.
2. Ensure that the **Enable Performance Monitoring Infrastructure (PMI)** check box is checked.
3. In the Currently Monitored Set box, click the **Custom** link.
4. In the Configuration window, expand the Workload Management tree and choose either **server**, if the application server is servicing EJB requests, or **client** if the application server is making EJB requests or both if applicable.
5. In the table on the right, select all of the available metrics and choose **Enable**.
6. Save your changes and restart the application server.

Figure 7-3 shows the workload management Client metrics enabled.



Select	Counter	Type	Description	Status
☐	ClientClusterUpdateCount	CountStatistic	The number of times that this client has received new server cluster information.	Enabled
☐	ClientResponseTime	TimeStatistic	The response time, in milliseconds, for IIOP requests sent by this client.	Enabled
☐	OutgoingIIOPRequestCount	CountStatistic	The number of outgoing IIOP requests sent from this client.	Enabled
Total 3				

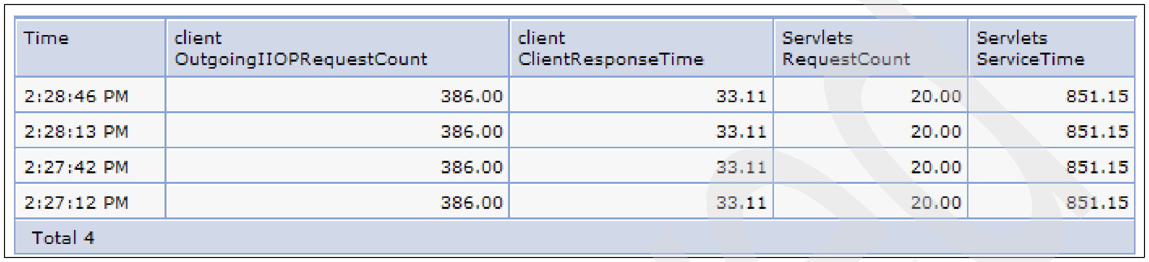
Figure 7-3 Enabling WLM PMI statistics

Recreate the issue and then review the statistics on each server that is acting as a client:

1. Navigate to **Monitoring and tuning** → **Performance viewer** → **Current activity** and click each server.

2. In the TPV window expand the `Performance modules` and then the `Workload management` trees.
3. Select the checkbox next to `client` and select **`View modules`**. You can then view the data as a graph or as a table.

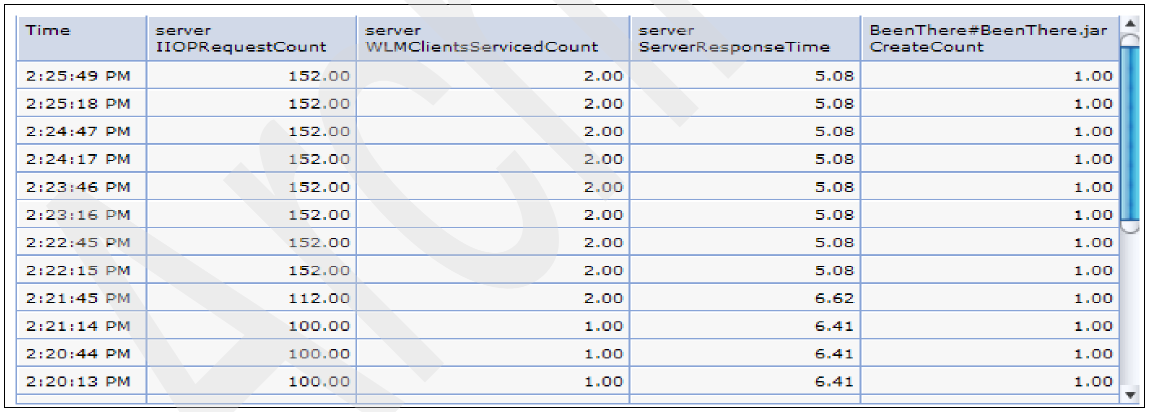
Figure 7-4 shows the EJB client request counts being made from a single servlet client to a cluster of two EJB servers. You can see that 386 requests were made.



Time	client OutgoingIIOPRequestCount	client ClientResponseTime	Servlets RequestCount	Servlets ServiceTime
2:28:46 PM	386.00	33.11	20.00	851.15
2:28:13 PM	386.00	33.11	20.00	851.15
2:27:42 PM	386.00	33.11	20.00	851.15
2:27:12 PM	386.00	33.11	20.00	851.15
Total 4				

Figure 7-4 EJB WLM client requests

Next, look at the EJB server statistics for the servers running the EJBs. Figure 7-5 shows the number of requests processed by one of the two servers. You can see that it processed 152 requests, this is approximately 40% of the client requests and represents reasonably even workload balancing.



Time	server IIOPRequestCount	server WLMClientsServicedCount	server ServerResponseTime	BeenThere#BeenThere.jar CreateCount
2:25:49 PM	152.00	2.00	5.08	1.00
2:25:18 PM	152.00	2.00	5.08	1.00
2:24:47 PM	152.00	2.00	5.08	1.00
2:24:17 PM	152.00	2.00	5.08	1.00
2:23:46 PM	152.00	2.00	5.08	1.00
2:23:16 PM	152.00	2.00	5.08	1.00
2:22:45 PM	152.00	2.00	5.08	1.00
2:22:15 PM	152.00	2.00	5.08	1.00
2:21:45 PM	112.00	2.00	6.62	1.00
2:21:14 PM	100.00	1.00	6.41	1.00
2:20:44 PM	100.00	1.00	6.41	1.00
2:20:13 PM	100.00	1.00	6.41	1.00

Figure 7-5 EJB WLM server requests

Note also the `WLMClientsServicedCount` statistic. This shows the number of clients that are making WLM based EJB requests. In this case there are two clients.

If you are seeing uneven load distribution across the cluster members, it may be caused by:

- Unbalanced cluster member weights. For more information, see “Unbalanced cluster member weights” on page 279.
- Transaction affinity causing request distribution to be uneven. For more information, see “Transaction affinity is skewing distribution” on page 281.
- The high availability manager may be disabled for a server. For more information, see “HA Manager is disabled” on page 282.

If you are still seeing unexpected distribution, first consider that the WLM uses a weighted proportional scheme to distribute EJB requests and uses feedback mechanisms that can change the routing behavior on the fly. It reacts to various scenarios and clustered server load when making routing decisions, so it is possible that WLM can function perfectly and the requests will not be balanced exactly as you expect.

If you still believe you are experiencing an uneven load balancing problem, review the JVM logs for any errors that might be causing a server to have problems processing requests.

Analyze the JVM logs

Look for the following:

- `com.omg.CORBA` errors.

  If you find any of these errors, go to “CORBA errors” on page 274.
- Look for `DCSV8050I` messages that show your application server is not in a core group view.

  If you see this message, go to “EJB server not in core group view” on page 282.

7.4.3 Root causes

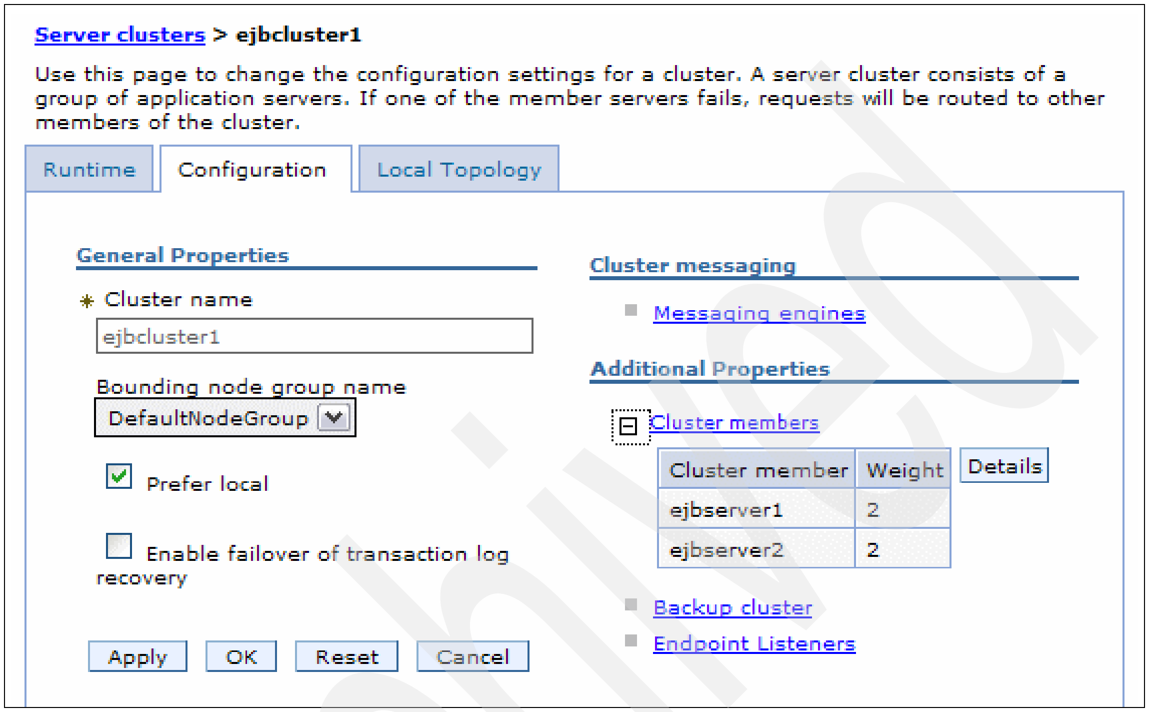
The following are possible root causes for this problem:

- Unbalanced cluster member weights
- Transaction affinity is skewing distribution
- HA Manager is disabled
- EJB server not in core group view

7.4.4 Unbalanced cluster member weights

Check the cluster member weights for the servers. These can be viewed in the administrative console as shown in Figure 7-6.

Navigate to **Servers** → **Clusters** → *clustername* and expand the `Cluster members` tree. The **Details** button allows you to change either the configured or runtime weights of each server in the cluster.



Server clusters > **ejbcluster1**

Use this page to change the configuration settings for a cluster. A server cluster consists of a group of application servers. If one of the member servers fails, requests will be routed to other members of the cluster.

General Properties

* Cluster name: ejbcluster1

Bounding node group name: DefaultNodeGroup

✓ Prefer local

☐ Enable failover of transaction log recovery

Cluster messaging

- Messaging engines

Additional Properties

Cluster members

Cluster member	Weight
ejbserver1	2
ejbserver2	2

- Backup cluster
- Endpoint Listeners

Figure 7-6 EJB cluster member weights

When you modify server weights you will impact the distribution of EJB requests among the servers. For example: if you have two servers and one is more powerful than the other, you might choose to assign server weights of 7 and 2 respectively. This means that server A will be sent 7 requests to process for every 2 that are sent to server B.

EJB WLM includes “fairness” balancing so that server weights of 2 and 7 will result in the 2:7 distribution ratio with pattern like:

```
AAAA-B-AAA-B
```

Rather than a pattern like:

```
A-B-A-B-AAAAA
```

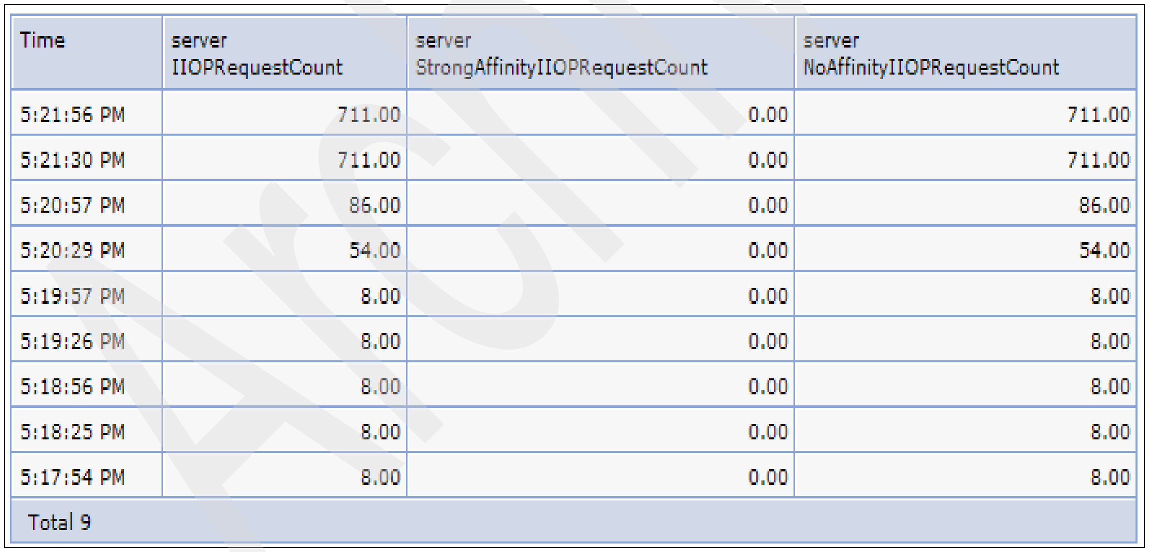
Resolve the problem

Ensure that the cluster member load balancing weights are equal or set as expected for the cluster.

7.4.5 Transaction affinity is skewing distribution

WebSphere Application Server EJB WLM supports transactional affinity. Transactional affinity means that a particular server must service all requests that comprise a particular transaction. A transaction is initiated and controlled by the client program making the EJB requests. If the application has been written so that multiple EJB requests are part of a single transaction and that transactional affinity is required, then the WLM will route requests that make up a transaction to the server that serviced the first request.

If there are a large number of requests involved in transactional affinity, this can cause uneven routing of requests. The WLM PMI data will show you how many requests were routed to a particular server due to transactional affinity. Figure 7-7 shows you the PMI data where there is no transaction affinity. Look for the `StrongAffinityIIOPRequestCount` field. In this example there is no transaction affinity in use.



Time	server IIOPRequestCount	server StrongAffinityIIOPRequestCount	server NoAffinityIIOPRequestCount
5:21:56 PM	711.00	0.00	711.00
5:21:30 PM	711.00	0.00	711.00
5:20:57 PM	86.00	0.00	86.00
5:20:29 PM	54.00	0.00	54.00
5:19:57 PM	8.00	0.00	8.00
5:19:26 PM	8.00	0.00	8.00
5:18:56 PM	8.00	0.00	8.00
5:18:25 PM	8.00	0.00	8.00
5:17:54 PM	8.00	0.00	8.00
Total 9			

Figure 7-7 PMI data for Transaction Affinity requests

Resolve the problem

As this is not really a problem with the workload manager but rather working as designed, there is no simple resolution. Review your application design so that you are not making large numbers of transactional affinity EJB requests.

> **Note:** This situation is likely to only occur in a test environment and will probably resolve itself as the number of distinct users and therefore unique transactions increases.

7.4.6 HA Manager is disabled

A server can be excluded from participating in the WebSphere Application Server High Availability service. This can be done if you have decided a server or cluster does not need the services provided by the HA manager. However disabling it causes WLM to function improperly. The HA Manager is enabled in a WebSphere Application Server environment by default, however it is possible to disable it if you have determined you do not need its functionality.

Resolve the problem

Enable the HA manager service for the server and the cluster. See the following WebSphere Information Center articles for more information:

- *When to use a high availability manager*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_ha_ham_required.html`

- *Disabling or enabling a high availability manager*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/trun_ha_ham_enable.html`

7.4.7 EJB server not in core group view

If the HA Manager is experiencing a problem where it can not determine the status of a cluster member, it will exclude that server from its core group view. If the HA Manager has excluded a server from the core group view, then the server is isolated from the rest of the cell and will not participate in EJB workload management.

Resolve the problem

Make sure the server is "in view" in reference to HA Manager and the core group the server belongs to. If the server is not in view then it may be isolated from the rest of the cell and not seen as being available by the other servers, and subsequently the client.

There are many reasons why this might occur. For more information, see 8.2.6, “Network issues causing split views” on page 299.

7.4.8 Validate the solution

Recreate the situation and validate that requests are being distributed to the clustered servers as you expect.

If you still have problems, go to “The next step” on page 287.

7.5 Failing server still receives EJB requests (failover fails)

This section discusses steps to diagnose your problem if you are seeing that the WLM is trying to send requests to a server despite that server being down.

7.5.1 Collect diagnostics

Review the JVM logs for both the client making the request and the server that is failing or has failed.

Collect the following for both client and server:

- JVM SystemOut and SystemErr logs

  See 9.1, “Collecting JVM logs” on page 314.

7.5.2 Analyze diagnostics

First review the JVM logs for the client making the EJB request. If this does not show you the root cause, then you might be able to determine what is happening from the JVM logs on the server that is failing or has failed.

JVM logs

Look for the following:

- CORBA messages indicating transaction retries or rollbacks such as:

```
org.omg.CORBA.TRANSIENT: SIGNAL_RETRY
org.omg.CORBA::TRANSACTION_ROLLEDBACK
```

If you see either of these, go to “In-flight transaction trying to complete” on page 284.

If the server that the EJB client is attempting to execute the transaction against is down and the EJB request can not be completed, you will see CORBA errors as described in “CORBA errors” on page 274. If you have other available cluster members that should be able to service the request then you will need to take an EJB WLM trace (see “Enabling the WLM trace” on page 314).

7.5.3 Root causes

The following is a possible root cause for this issue:

- In-flight transaction trying to complete

7.5.4 In-flight transaction trying to complete

The client might have been in a running transaction with an EJB on the server that went down.

Since the transaction might have completed, failing over this request to another server could result in this request being serviced twice. Therefore the WLM puts the cluster member into Quiesce mode. While in Quiesce mode, the server will reject all incoming requests which it determines are new work, but still allow in-flight requests to complete. This is primarily designed to allow transaction work to finish as above to prevent unnecessary TRANSACTION ROLLBACK exceptions.

By default, Quiesce mode will last for a maximum of 3 minutes (this is configurable), although it is possible for a server to exit quiesce earlier if all registered components agree that it is okay to do so based on their own criteria.

Resolve the problem

If a request gets rejected by a server in Quiesce mode, WLM will be called with an `org.omg.CORBA.COMM_FAILURE` with a completion status of "`no`", and this request will be automatically retried by the WLM.

It might be that no servers are available to process the request. Refer to “CORBA errors” on page 274.

7.5.5 Validate the solution

Recreate the situation and validate that requests are being distributed to the clustered servers as you expect.

If you still have problems, go to “The next step” on page 287.

7.6 Restarted servers do not share the workload

This error occurs when the servers that were unavailable are not recognized by the workload manager component after they are restarted.

7.6.1 Collect diagnostics

You will need to review the JVM logs from the EJB client and the EJB server you have restarted. Collect the following for both client and server:

- JVM SystemOut and SystemErr logs

  See 9.1, “Collecting JVM logs” on page 314.

7.6.2 Analyze diagnostics

Review the JVM SystemOut and SystemErr logs for the restored EJB server.

JVM logs

Look for the following:

- Ensure the server has finished starting, look for the message that indicates a server is ready to process requests:

  ```
  [3/05/07 17:01:21:959 EST] 0000000a WsServerImpl  A   WSVR0001I:
  Server ejbserver1 open for e-business
  ```

  If this message has not appeared, either wait for the server to finish startup or try restarting it again.

- Ensure the EJB has started correctly:

  If the EJB fails to start, you will exceptions that will describe what has gone wrong with the application on startup. Example 7-3 shows an EJB failing to start due to a `ClassNotFoundException`.

Example 7-3   EJB fails to start

```
[3/05/07 20:50:29:517 EST] 00000039 EJBContainerI E   WSVR0068E: Attempt to
start EnterpriseBean BeenThere#BeenThere.jar#BeenThereBean failed with
exception: com.ibm.ejs.container.ContainerException: Failed to initialize
BeanMetaData instance; nested exception is:
   java.lang.ClassNotFoundException:
com.ibm.websphere.samples.beenthere.EJSStatelessBeenThereBeanHomeBean_271d3519
```

7.6.3 Root causes

Some possible root causes for this issue are:

- Server has not started, wait for the server to start or retry server startup
- Unusable interval has not expired

7.6.4 Unusable interval has not expired

Once the WLM has marked a server as unavailable, it waits for a period of time before trying to send a request to that server. This is called the unusable interval and it is set to 300 seconds (or 5 minutes) by default.

You can confirm that this is the problem by ensuring that the server is up and running and then wait for the unusable interval to elapse before checking to determine whether load balancing occurs as described in “Analyze the PMI data” on page 277.

Resolve the problem

You may need to either wait longer for the server to begin processing requests, or decrease the unusable interval.

You can adjust the unusable interval by setting the custom property `com.ibm.websphere.wlm.unusable.interval` to a value more suitable to your environment. The parameter is set in seconds.

This parameter is passed to the JVM as a “-D” parameter. If you are using the standalone WebSphere or Java application client as your EJB client, modify the command line to include this parameter:

```
java -Dcom.ibm.websphere.wlm.unusable.interval=150 MyApplication
```

If your client is an application server, you set this parameter on the JVM properties page. Navigate to **Servers** → **Application Servers** → *server_name* → **Java and process management** → **Process definition** → **Java Virtual Machine**. Scroll down to **Generic JVM arguments** and enter the parameter in the text box. Save your changes. You will need to restart the server for this change to take effect.

7.6.5 Validate the solution

Stop and restart the server to see if the WLM will start routing requests to the server.

If you still have problems, go to “The next step” on page 287.

7.7 The next step

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, there are other things that can go wrong during with the EJB workload management.

Search online support

If you are sure the problem is with EJB WLM, there are things that you can do before contacting IBM support. First, you should review the documentation that you have gathered for errors that were not addressed in this paper and search support sites for information or fixes. Look for current information available from IBM support on known issues and resolutions on the following IBM support page using the following argument:

http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCMPEP

Look also at the WebSphere Information Center documentation for additional resources for diagnosing and fixing EJB workload management issues:

- *Clusters and workload management*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_srvgrp.html

- *Balancing workloads with clusters*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm.html

- *Tuning a workload management configuration*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm_tuning.html

Re-evaluate the symptoms

If, after going through this process, you still have an undiagnosed problem, we recommend that you go back to *Approach to Problem Determination in WebSphere Application Server V6* at:

http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf

Review the problem classifications to see if there are any other components that might be causing the problem.

Contact IBM

If these steps do not resolve your problem, then gather additional information as specified in the following MustGather document and raise a problem record with IBM. The following URL contains a list of the MustGather documentation for EJB workload management problems.

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21052165

8

High availability manager problem determination

The *high availability manager* (commonly called the *HA manager*) is a component of WebSphere Application Server that enhances the availability of WebSphere Application Server singleton services such as transaction or messaging services. It provides a peer recovery mechanism for in-flight transactions or messages among clustered application servers. It also supports memory-to-memory session replication.

The HA Manager runs as a service within each WebSphere process; deployment manager, node agents, and application servers. In the event of a server failure, the HA Manager will failover any singleton service that was running on the failed server to a peer server. Examples of such a failover include the recovery of any in-flight transactions or the restarting of any messaging engines that were running on the failed server.

Problems with the HA Manager are typically identified by HA Manager related messages in the application server logs. However, they can also manifest as singleton services not being failed over when required or as failure to complete in-flight transactions from a failed server.

8.1 Determine if you need to use the HA Manager

The HA Manager consumes system resources, such as CPU cycles, heap memory, and sockets.

A *core group* is a high availability domain within a cell. It serves as a physical grouping of JVMs in a cell that are candidates to host singleton services. It can contain stand-alone servers, cluster members, node agents, or the deployment manager. Each of these run in a separate JVM. The amount of resources that the HA Manager consumes increases exponentially as the size of a core group increases. For large core groups, the amount of resources that the HA Manager consumes can become significant. If these services are not used then you can disable the HA Manager and free these resources.

You can disable the HA Manager on a given application server process if you do not use any of the following services:

- Memory-to-memory replication
- Singleton failover
- Workload management routing
- On-demand configuration routing

Do not disable the HA Manager on any administrative process (node agent or the deployment manager) unless the HA Manager is disabled on all application server processes in that core group.

If you disable the HA Manager on one member of a cluster, you must disable it on all of the other members of that cluster.

The following WebSphere Information Center article describes this in detail.

- *When to use a high availability manager*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_ha_ham_required.html

The following WebSphere Information Center article describes how to disable the HA Manager.

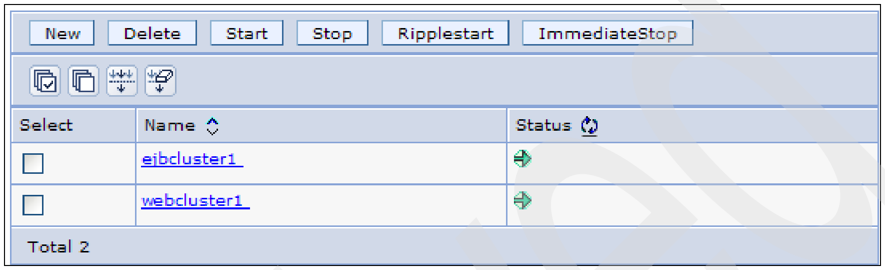
- *Disabling or enabling a high availability manager*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_ha_ham_enable.html

8.1.1 Check system integrity

If you think you might have a problem with the HA Manager, the first step in diagnosing the problem is to verify the system integrity:

1. In the administrative console, navigate to **`Servers`** → **`Clusters`** and ensure your clusters are started (as shown in Figure 8-1). All cluster members need to be started to ensure even load balancing.



Select	Name	Status
☐	ejbcluster1	➡
☐	webcluster1	➡

Total 2

Figure 8-1 Started clusters

2. Ensure that all the machines in your configuration have TCP/IP connectivity to each other by running the **`ping`** or **`telnet`** command:
   - From each physical server to the deployment manager
   - From the deployment manager to each physical server
   - From the client to each physical server
   - Between all physical servers
3. Ensure your core group size is not excessive.

   The default core group that is created when you create a cell is made of all member servers in that cell including node agents and the deployment manager.

   IBM recommends that you limit your core group size to 50 members for performance reasons. If you have large numbers of cell members, you will need to split up your core groups. For more information about this, see "Excessive core group sizes" on page 301.

8.1.2 Identify symptoms

The following are typical symptoms of an HA Manager problem.

- Singleton services not starting on failover
- Singleton services starting unexpectedly
- Server will not start

- Excessive resource usage
- HA Manager messages in the logs. These will be prefixed with HMGR, CWRLS or DCSV.

8.2 Singleton services not starting on failover

This section discusses steps to diagnose your problem if you have configured the system with a cluster of services and are using one or more singleton services, such as a messaging engine or transaction log recovery. However, you find that on server failure, these services are not being failed over.

8.2.1 Collect diagnostics

Collect the following:

- JVM SystemOut and SystemErr logs

You may need to review logs from more than just the application server that is reporting the problems. The High Availability service is cluster-aware so problems can be spread across the cluster.

Depending on the problem you are investigating, you should consider simplifying the problem determination process by reducing the number of servers to monitor while debugging the problem. You would do this by shutting down all but the minimum number of servers required to recreate the issue.

8.2.2 Analyze diagnostics

First look in the JVM logs for the messages related to servers in the cluster other than the server that has failed. You are looking for messages related to the processing of the HA manager and for a reason why it has not detected a failed server.

Look for the following:

- `HMGR0140E - An event indicating that the core group membership is inconsistent was received. Recovery failed. The exception is {0}.`

  Go to "Inconsistent group membership" on page 293.
- `HMGR1001W - An attempt to receive a message of type {0} for Agent {1} in AgentClass {2} failed. The exception is {3}`

  Go to "Thread pool problem" on page 299.

- `DCSV8050I messages that report inconsistent view sizes across cluster members`

  Go to “Network issues causing split views” on page 299.

8.2.3 Root causes

The following are possible root causes for HA Manager problems:

- Inconsistent group membership
- Thread pool problem
- Network issues causing split views

8.2.4 Inconsistent group membership

Inconsistent core group membership means that the HA Manager’s view of the servers available in the core group is different from the actual servers that are available. This can occur if changes are made to the configuration that are not fully synchronized when servers are started.

Compare the core group views to the cluster topology.

Verify the size and status of the core group

Verify the size and the status of the core group by reviewing the messages in the application server logs from the member servers. One or more servers, depending on your configuration, will be elected as the *core group coordinator*. The core group coordinator is the service that monitors and maintains the core groups configured to run in the HA cluster. Example 8-1 shows the `HMGR0206I` message to indicate a server is the core group coordinator.

Example 8-1 Core group messages

```
[30/04/07 14:43:56:627 EST] 00000024 CoordinatorIm I   HMGR0206I: The
Coordinator is an Active Coordinator for core group DefaultCoreGroup.
...
[30/04/07 14:45:05:446 EST] 00000024 CoordinatorIm I   HMGR0207I: The
Coordinator was previously an Active Coordinator for core group
DefaultCoreGroup but has lost leadership.
[30/04/07 14:45:05:446 EST] 00000024 CoordinatorIm I   HMGR0218I: A new core
group view has been installed. The core group is DefaultCoreGroup. The view
identifier is (8:0.IBM99TVXRDCell1\IBM99TVXRDCellManager1\dmgr). The number of
members in the new view is 7.
[30/04/07 14:45:05:556 EST] 00000024 CoreGroupMemb I   DCSV8050I: DCS Stack
DefaultCoreGroup at Member IBM99TVXRDCell1\IBM99TVXRDNode2\nodeagent: New view
```

```
installed, identifier (8:0.IBM99TVXRDCell1\IBM99TVXRDCellManager1\dmgr), view
size is 7 (AV=7, CD=7, CN=7, DF=7)
[30/04/07 14:45:05:827 EST] 00000018 ViewReceiver  I   DCSV1033I: DCS Stack
DefaultCoreGroup at Member IBM99TVXRDCell1\IBM99TVXRDNode2\nodeagent: Confirmed
all new view members in view identifier
(8:0.IBM99TVXRDCell1\IBM99TVXRDCellManager1\dmgr). View channel type is
View|Ptp.
```

Example 8-1 also shows the HMGR0207I message that shows a server was a core group coordinator but has lost the leadership of the core group. Finally, it shows the messages that are produced when a new core group view is installed.

The DCSV8050I message contains the following information:

- AV = Number of Active members in the view
- CN = Members which are connected to this member. Usually, this should be the same as AV.
- CD = CN - number of suspect members

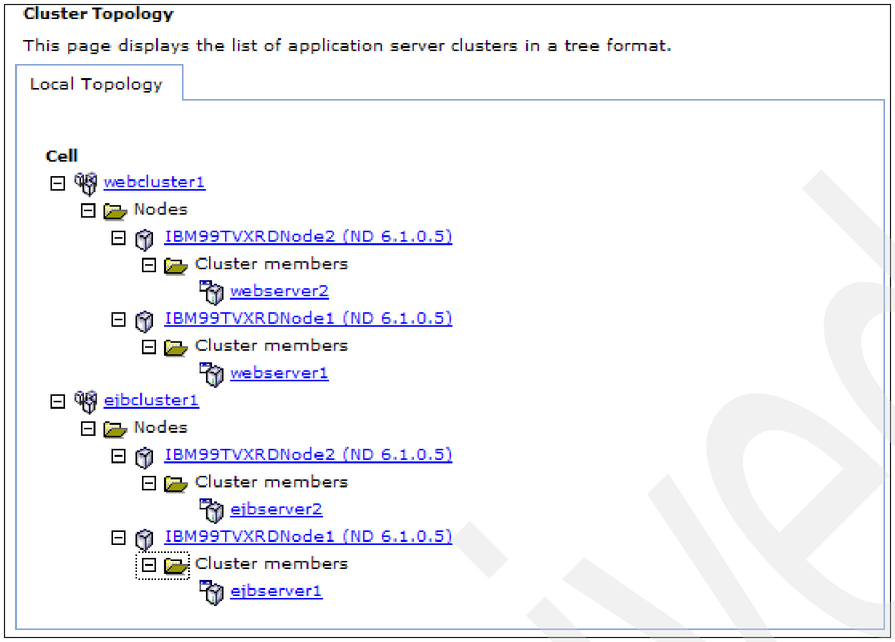
  That is the number of members that the server has a connection to, but the HA Manager suspects their integrity and does not want to be in the same view with them.
- DF = Number of defined members.

If all is running as expected and all core group servers are started, then the number of active members (AV) will equal the number of defined members (DF)

A new core group view is installed every time a server that is part of the core group is stopped or started. Installing a new view might result in significant, temporary spikes in the amount of CPU consumed and the network bandwidth used.

Verify the cluster topology

Check to ensure that the cluster topologies are what you expect in the administrative console. You can review the cluster topology in the administrative console as shown in Figure 8-2. Navigate to **Servers** → **Cluster topology**. Compare this to the list of servers that the core group coordinator is aware of as shown in Figure 8-3 on page 296. Ensure that they both list the same servers.


Cluster Topology

This page displays the list of application server clusters in a tree format.

Local Topology

```
Cell
⊟ webcluster1
   ⊟ Nodes
      ⊟ IBM99TVXRDNode2 (ND 6.1.0.5)
         ⊟ Cluster members
            webserver2
      ⊟ IBM99TVXRDNode1 (ND 6.1.0.5)
         ⊟ Cluster members
            webserver1
⊟ ejbcluster1
   ⊟ Nodes
      ⊟ IBM99TVXRDNode2 (ND 6.1.0.5)
         ⊟ Cluster members
            ejbserver2
      ⊟ IBM99TVXRDNode1 (ND 6.1.0.5)
         ⊟ Cluster members
            ejbserver1
```

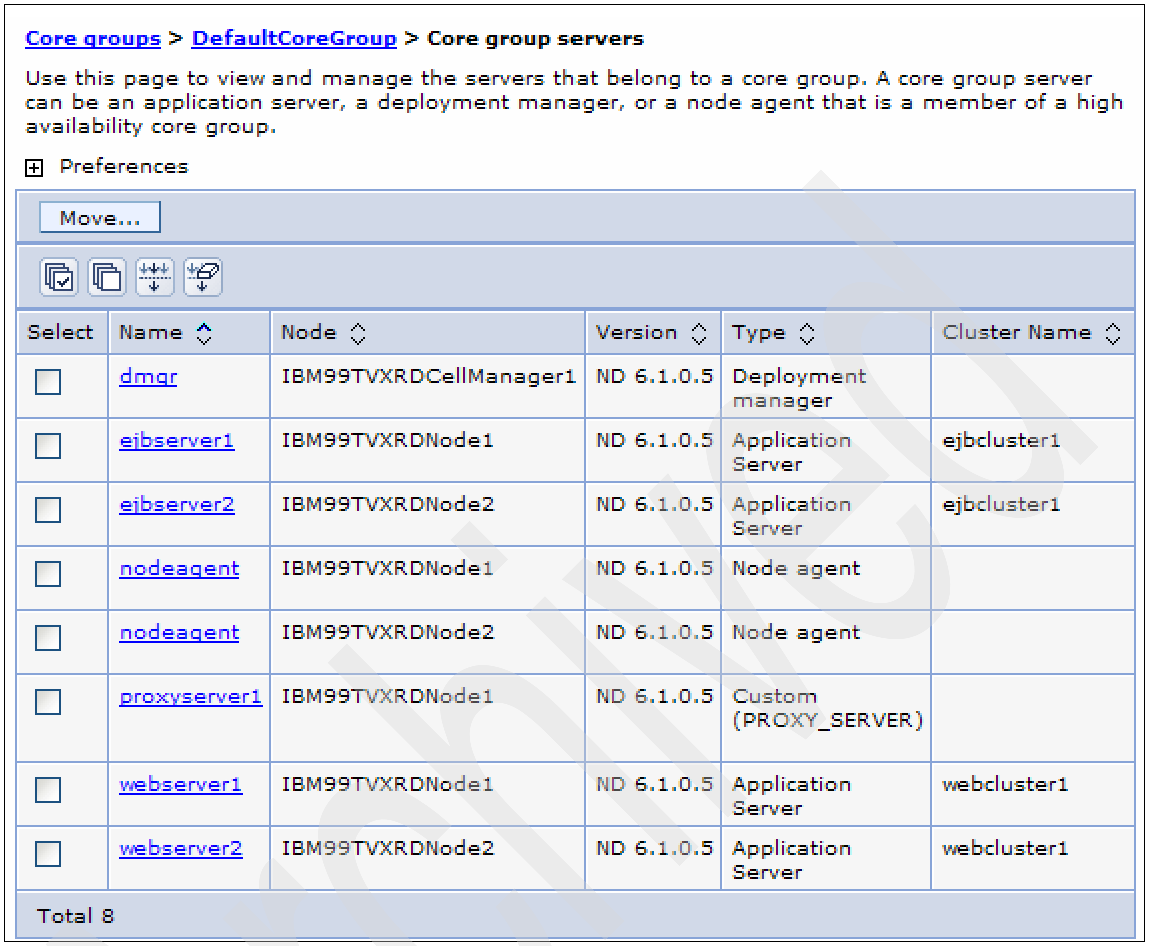
Figure 8-2 Cluster topology

Verify the HA Manager configuration

You can view the configuration of the HA Manager using the administrative console.

Navigate to **`Servers`** → **`Core groups`** → **`Core group settings`** → *core_group_name* → **`Core group servers`**.

Figure 8-3 shows the servers that the HA Manager expects to be in the core group. That is, the list of servers that the core group coordinator is aware of and connected to. Each of these servers will be a member of one or more core groups. Each core group has its own core group view, that is a view of the servers participating in the core group.


Core groups > DefaultCoreGroup > Core group servers

Use this page to view and manage the servers that belong to a core group. A core group server can be an application server, a deployment manager, or a node agent that is a member of a high availability core group.

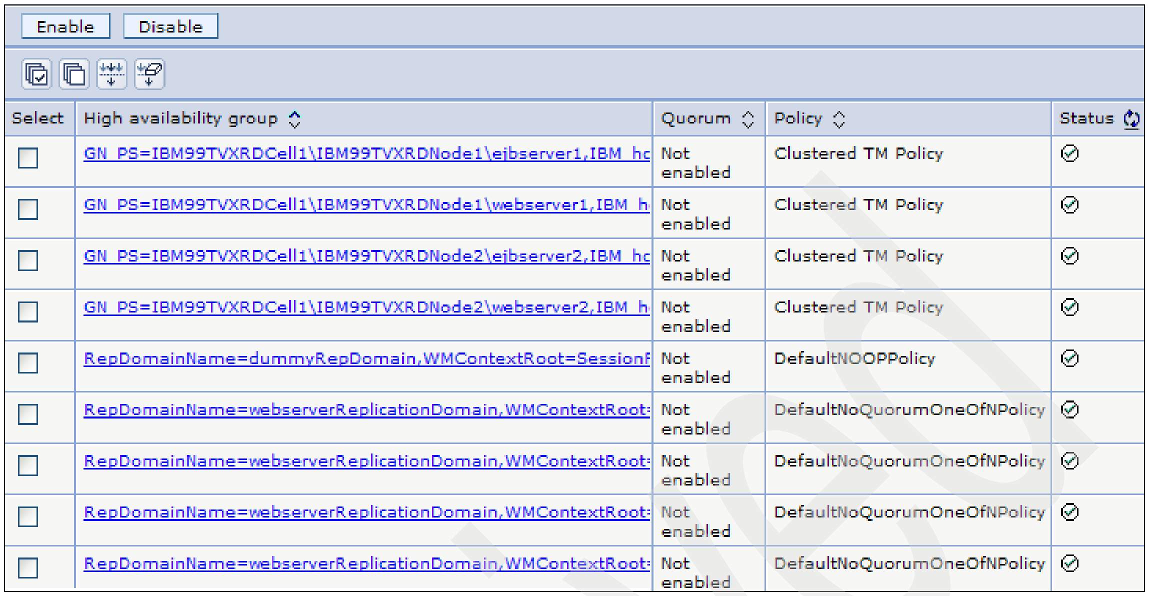
Select	Name	Node	Version	Type	Cluster Name
☐	dmgr	IBM99TVXRDCellManager1	ND 6.1.0.5	Deployment manager	
☐	ejbserver1	IBM99TVXRDNode1	ND 6.1.0.5	Application Server	ejbcluster1
☐	ejbserver2	IBM99TVXRDNode2	ND 6.1.0.5	Application Server	ejbcluster1
☐	nodeagent	IBM99TVXRDNode1	ND 6.1.0.5	Node agent	
☐	nodeagent	IBM99TVXRDNode2	ND 6.1.0.5	Node agent	
☐	proxyserver1	IBM99TVXRDNode1	ND 6.1.0.5	Custom (PROXY_SERVER)	
☐	webserver1	IBM99TVXRDNode1	ND 6.1.0.5	Application Server	webcluster1
☐	webserver2	IBM99TVXRDNode2	ND 6.1.0.5	Application Server	webcluster1

Total 8

Figure 8-3 Core group members

You can also review the core groups that are running at any given point in time from the administrative console.

Navigate to **Servers** → **Core groups** → **> Core group settings** → *core_group_name* and click the **Runtime** tab. Then choose **Show groups**. Figure 8-4 shows the listing of the active core groups in the cell. These are also known as `High availability groups`.



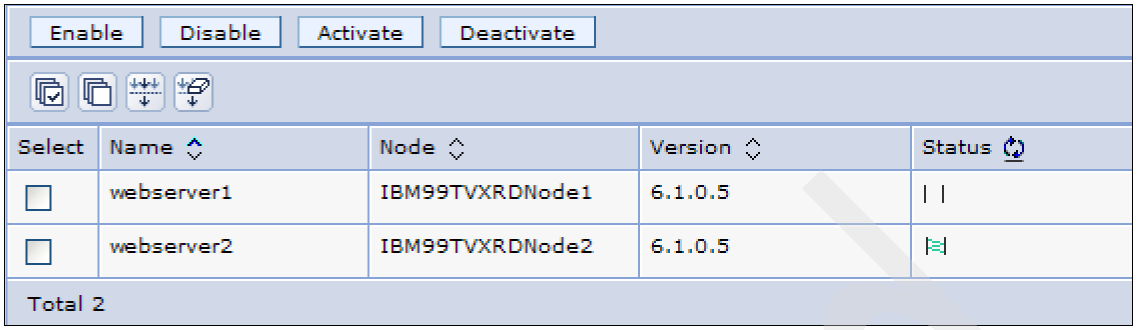
Select	High availability group	Quorum	Policy	Status
☐	GN_PS=IBM99TVXRDCell1\IBM99TVXRDNode1\ejbserver1,IBM_hc	Not enabled	Clustered TM Policy	✓
☐	GN_PS=IBM99TVXRDCell1\IBM99TVXRDNode1\webserver1,IBM_h	Not enabled	Clustered TM Policy	✓
☐	GN_PS=IBM99TVXRDCell1\IBM99TVXRDNode2\ejbserver2,IBM_hc	Not enabled	Clustered TM Policy	✓
☐	GN_PS=IBM99TVXRDCell1\IBM99TVXRDNode2\webserver2,IBM_h	Not enabled	Clustered TM Policy	✓
☐	RepDomainName=dummyRepDomain,WMContextRoot=SessionF	Not enabled	DefaultNOOPPolicy	✓
☐	RepDomainName=webserverReplicationDomain,WMContextRoot:	Not enabled	DefaultNoQuorumOneOfNPolicy	✓
☐	RepDomainName=webserverReplicationDomain,WMContextRoot:	Not enabled	DefaultNoQuorumOneOfNPolicy	✓
☐	RepDomainName=webserverReplicationDomain,WMContextRoot:	Not enabled	DefaultNoQuorumOneOfNPolicy	✓
☐	RepDomainName=webserverReplicationDomain,WMContextRoot:	Not enabled	DefaultNoQuorumOneOfNPolicy	✓

Figure 8-4 Active core groups

Note that core groups or servers will only appear in the Runtime tab when the associated server or servers are running.

You will typically see a lot of core groups in the Runtime tab. For example, if you have enabled HTTP session replication and have installed six Web applications, that is, with six separate context roots, then you will see six separate core groups. You will also see one core group for each server to manage the session replication cache. These will be in addition to any other core groups to manage the other singleton services.

If you click a core group name, you will see the servers that are part of that particular core group as shown in Figure 8-5. This is known as the core group view.



Select	Name	Node	Version	Status
☐	webserver1	IBM99TVXRDNode1	6.1.0.5	| |
☐	webserver2	IBM99TVXRDNode2	6.1.0.5	⊨

Total 2

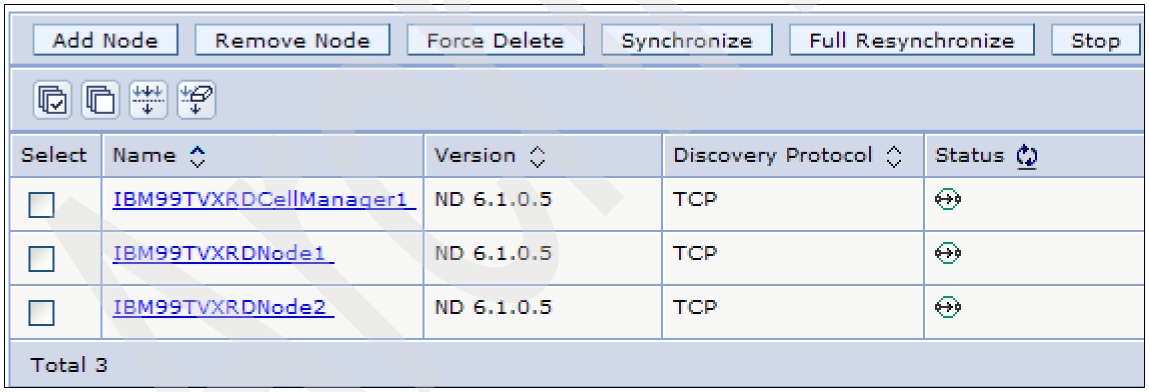
Figure 8-5 Core group view

Resolve the problem

Differences between the core group views and the cluster topology can be caused by corruptions in the WebSphere Application Server configuration repository. Try the following steps to restore the configuration:

1. Perform a full resynchronization from the deployment manager to all nodes.

   In the administrative console, navigate to **System Administration** → **Nodes**. Check the `Select` check box for all nodes except the manager node as shown in Figure 8-6. Click the **Full Resynchronize** button.



Select	Name	Version	Discovery Protocol	Status
☐	IBM99TVXRDCellManager1	ND 6.1.0.5	TCP	⇔
☐	IBM99TVXRDNode1	ND 6.1.0.5	TCP	⇔
☐	IBM99TVXRDNode2	ND 6.1.0.5	TCP	⇔

Total 3

Figure 8-6 Full resynchronize

2. Restart all servers in the cell.
3. If the previous steps do not resolve the problem, restore the configuration from a known good backup.

   WebSphere Application Server supplies the following command line tools for the backup and restore of a configuration:

   – `backupConfig.bat or backupConfig.sh`

– `restoreConfig.bat or restoreConfig.sh`

8.2.5 Thread pool problem

If the HA Manager is unable to obtain a communications thread from the default thread pool, you will see the `HMGR1001W`. This will generally be accompanied by messages showing view sizes changing unexpectedly. Among the messages you may find the following:

```
[8/26/06 5:18:33:113 EDT] 00000095 WorkQueueMana W   TCPC0005W: TCP
Channel channel_0 could not obtain thread from thread pool <null>.
```

This problem is due to a Data Replication Service (DRS) issue, where the DRS processes are consuming all the threads from the default thread pool. As a result, the transport threads used by the HA Manager (the DCS connections) are being closed unexpectedly. This leads to instability in the core group views which can lead to unexpected failures under error conditions and excessive amounts of system resources to be consumed.

Resolve the problem

Create a separate thread pool for DCS communications so that it is isolated from the DRS communications threads.

Refer to the following technote for details:

- *Tune High Availability (HA) Manager configuration for large cell environments*

  `http://www.ibm.com/support/docview.wss?rs=180&uid=swg21251873`

8.2.6 Network issues causing split views

If you are seeing inconsistent view sizes across the servers in your cluster, then you are experiencing a communications issue between the cluster members.

Consider a cluster with seven defined members including the deployment manager, two node agents and four application servers. The view size in this environment would be 7 and so a normal core group view message for when all servers are running would be as shown in Example 8-2.

Example 8-2   Normal core group view

```
[2/05/07 09:27:51:576 EST] 00000021 CoreGroupMemb I   DCSV8050I: DCS Stack
DefaultCoreGroup at Member IBM99TVXRDCell1\IBM99TVXRDNode1\server1: New view
installed, identifier (22:0.IBM99TVXRDCell1\IBM99TVXRDCellManager1\dmgr), view
size is 7 (AV=7, CD=7, CN=7, DF=7)
```

However, if you see messages similar to those shown in Example 8-3 from server1 and Example 8-4 from server2, then you are seeing inconsistent view sizes across the cluster members.

Example 8-3 Broken core group view from server1

```
[2/05/07 09:27:51:576 EST] 00000021 CoreGroupMemb I   DCSV8050I: DCS Stack
DefaultCoreGroup at Member IBM99TVXRDCell1\IBM99TVXRDNode1\server1: New view
installed, identifier (22:0.IBM99TVXRDCell1\IBM99TVXRDCellManager1\dmgr), view
size is 2 (AV=2, CD=2, CN=2, DF=7)
```

Example 8-4 Broken core group view from server2

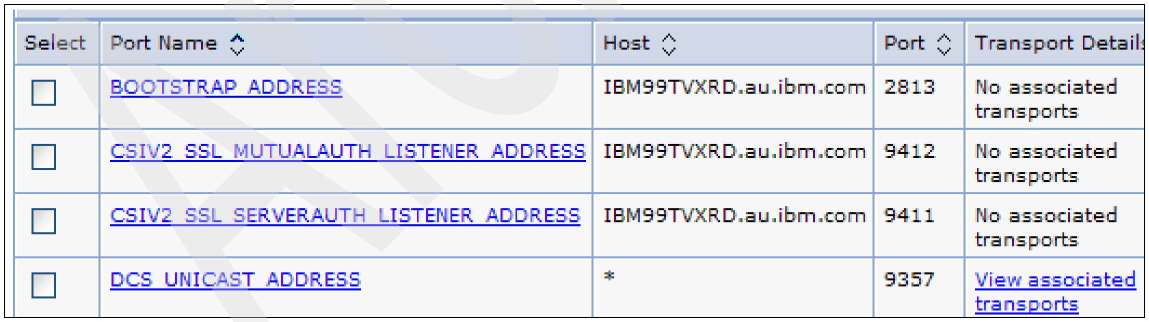
```
[2/05/07 09:27:51:576 EST] 00000021 CoreGroupMemb I   DCSV8050I: DCS Stack
DefaultCoreGroup at Member IBM99TVXRDCell1\IBM99TVXRDNode1\server2: New view
installed, identifier (22:0.IBM99TVXRDCell1\IBM99TVXRDCellManager1\dmgr), view
size is 3 (AV=3, CD=3, CN=3, DF=7)
```

Resolve the Problem

This problem is caused by a communications problem between the cluster members. Verify your communications as discussed in "Check system integrity" on page 291.

The HA Manager uses the Distributed Communications Service (DCS) for communications amongst the HA managers running in each process.

Using the loopback adapter can lead to this problem. By default, the DCS endpoint addresses use a `Host` field of "*" (asterisk) to indicate any host can respond on that particular port as shown in Figure 8-7.



Select	Port Name	Host	Port	Transport Detail
☐	BOOTSTRAP_ADDRESS	IBM99TVXRD.au.ibm.com	2813	No associated transports
☐	CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	IBM99TVXRD.au.ibm.com	9412	No associated transports
☐	CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	IBM99TVXRD.au.ibm.com	9411	No associated transports
☐	DCS_UNICAST_ADDRESS	*	9357	View associated transports

Figure 8-7 DCS address endpoint

When DCS resolves the host name of a machine that it is currently running on, in addition to the expected IP address of that host, it also gets back the loopback adapter address. This causes connectivity issues within the topology. The loopback adapter should be disabled or removed. However, if the loopback adapter is really needed, then the solution to this problem is to change the host field for each `DCS_UNICAST_ADDRESS` endpoint to use a dotted IP address rather than using '*' or host names.

All WebSphere Application Server processes will need to be synchronized and restarted after making these changes.

8.2.7 Excessive core group sizes

If your core group size is very large, you could find that the core group members will use more resources maintaining the core group than processing application requests. The amount of CPU and network usage increases exponentially as more members are added to a core group and this can lead to failures in the core group configuration data. Recovery from these failures is CPU intensive which may result in paging causing further failures.

Resolve the problem

Limit core group size to 50 members.

For large topologies, create multiple smaller core groups and link these core groups with a core group bridge. Core group bridges allow servers in separate core groups to share WLM information.

Refer to the WebSphere Information Center for instructions on creating and bridging multiple core groups:

- *Creating a new core group (high availability domain)*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/trun_ha_newcg.html

Validate the solution

After making the changes described in this section, you will need to resynchronize the changes and restart the services. Then recreate the issue and ensure the problem no longer occurs.

If you are still experiencing issues with singleton services not starting on failover, go to “The next step” on page 310.

8.3 Singleton services starting unexpectedly

This section discusses steps to diagnose your problem if you have configured the system with a cluster of services and are using one or more singleton services, such as a messaging engine or transaction log recovery. However, you find that on server failure, these services are not being failed over.

The reasons why you are seeing multiples of singleton services unexpectedly are likely to be the same as those causing singleton services not to start. Go to "Singleton services not starting on failover" on page 292 to resolve this issue.

Validate the solution

After making any changes you will need to resynchronize the changes and restart the servers. Then recreate the issue and ensure the problem no longer occurs.

If you are still experiencing issues with multiple occurrences of singleton services, go to "The next step" on page 310.

8.4 Server will not start

There are many reasons why an application server will not start. You will need to collect diagnostics to determine that the reason it is not starting is due to the HA Manager.

8.4.1 Collect diagnostics

Collect the JVM SystemOut and SystemErr logs.

8.4.2 Analyze diagnostics

If you do not see the normal "`open for e-business`" message associated with a started server and instead see the following message continuously logged:

```
CWRLS0030W: Waiting for HAManager to activate recovery
```

Go to "Unable to obtain lock on transaction log" on page 303.

The resolution of other startup failures are outside the scope of this document.

8.4.3 Root causes

The following is a possible root cause of the server not starting:

- Unable to obtain lock on transaction log

8.4.4 Unable to obtain lock on transaction log

This section discusses steps to diagnose your problem if the application server does not start and the following message is being logged continuously in the application server logs:

```
CWRLS0030W: Waiting for HAManager to activate recovery
```

Resolve the problem

The Transaction Manager relies on the HA Manager to assign it ownership of its transaction log file. This is true even if the "highly available transaction log" feature is not used. The Transaction Manager logs a `CWRLS0030W` message when it is waiting for the HA Manager to assign it ownership of its transaction log.

In order for the HA Manager to assign ownership of the transaction log to the Transaction Manager, the application server must establish itself as a member of a core group view. This requires the HA Manager to establish network connectivity between all running core group members.

Reference the following technote to resolve this issue:

- *CWRLS0030W message continuously logged and WebSphere Application Server fails to open for e-business*

  http://www.ibm.com/support/docview.wss?uid=swg21245012

8.4.5 Validate the solution

After making the changes described in this section, you will need to synchronize the changes and restart the services. Then recreate the issue and ensure the problem no longer occurs.

If you are still experiencing issues with singleton services not starting on failover, go to "The next step" on page 310.

8.5 Excessive resource usage

The core group coordinator runs on one or more services in the core group. Under some circumstances, the core group coordinator can lead to excessive resource usage.

8.5.1 Collect diagnostics

Review the JVM logs for the cluster members to determine where the core group coordinator is running. You may also need to review PMI statistics and server metrics to measure resource usage.

- JVM SystemOut and SystemErr logs
- Verbose GC data
- Server metrics

8.5.2 Analyze diagnostics

Analyze the JVM logs

First, identify the location of the core group coordinator from the JVM logs.

Look for the following in each server's JVM log:

- `HMGR0206I - The Coordinator is an Active Coordinator for core group DefaultCoreGroup.`

  This message indicates this server is a core group coordinator. Review the verbose GC data, PMI statistics and server metrics for this server.

- In addition, look for the following message in the logs:

  ```
  HMGR0152W - CPU Starvation detected. Current thread scheduling delay
  is {0} seconds.
  ```

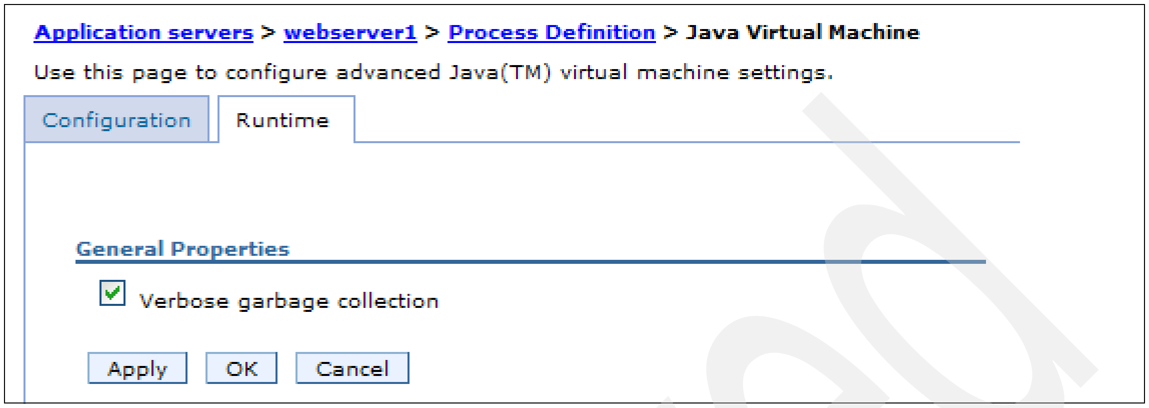
  This indicates a resource problem on the server. Go to "Too much load due to HA Manager" on page 307.

If you don't see any of these symptoms, continue with the problem analysis by looking at the verbose GC data.

Verbose GC data

If your JVM heap size usage is getting close to its maximum, you are likely to see an excessive number of garbage collections being performed. Frequent garbage collections in the JVM will increase the CPU being used on the server and reduce the number of requests that can be processed.

Verbose GC can be enabled dynamically on a running server through the administrative console, see Figure 8-8.


Application servers > webserver1 > Process Definition > Java Virtual Machine
Use this page to configure advanced Java(TM) virtual machine settings.
Configuration
Runtime
General Properties
Apply
OK
Cancel

[✓] Verbose garbage collection

Figure 8-8 Enable verbose GC in the administrative console

Look for the following:

- Excessive garbage collection.

  You can determine how frequently GC is running by looking at the `intervalms` value in the verbose GC data. This shows the number of milliseconds since the last run as shown in Example 8-5.

Example 8-5 verbose GC interval data

```
<af type="tenured" id="12" timestamp="Mon Apr 30 12:26:57 2007"
intervalms="2850.885">
```

  You can also look at verbose GC data using the IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT) tool that is run from the IBM Support Assistant (ISA) tool. You can use this tool to show you graphs of the GC activity.

- An OutOfMemory error

  This will appear in the verbose GC output and also in the application server logs. Look for the message shown in Example 8-6.

Example 8-6 OutOfMemory in application server logs

```
[30/04/07 17:00:07:893 EST] 0000001d ServletWrappe E   SRVE0068E: Uncaught
exception thrown in one of the service methods of the servlet: MyApplication. Exception
thrown : java.lang.OutOfMemoryError
```

  If you are seeing this occurring only in the cluster member running the core group coordinator, go to “High JVM heap usage” on page 306.

Note that the HA Manager is only one component that uses memory in a JVM, there may be other reasons outside of the scope of this document that are causing the out of memory condition.

If you don't see any of these symptoms, continue with the problem analysis by looking at the server metrics.

Server metrics

Server metrics such as CPU, I/O and page space utilization can also show uneven or unexpected load balancing. You should use the tool most appropriate to your operating system to check these statistics. For example, in Windows you could use **`Task Manager`** or **`Performance Monitor`** while in a Unix or Linux environment, you could use **`vmstat`** or **`top`**.

Look for the following:

- Unexpectedly high CPU, I/O or page space usage on the server running the HA Manager.

  If you see this condition, go to "Too much load due to HA Manager" on page 307.

8.5.3 Root causes

Some possible roots causes for this issue are:

- High JVM heap usage
- Too much load due to HA Manager

8.5.4 High JVM heap usage

The HA Manager process, and in particular, the core group coordinator uses extra resources on the application servers, node agents and deployment manager. If your JVM heap size is getting close to the limit even without considering the HA Manager components, then you could experience OutOfMemory error conditions when the HA Manager components start doing work.

Resolve the problem

If possible, increase the maximum JVM heap size on the server process to provide more heap memory for the components to use.

1. In the administrative console navigate to the process:
   - For an application server,

     `Servers` → **`Application servers`** → *server_name*
   - For a node agent,

     `System administration` → **`Node agent`** → *nodeagent*
   - For the deployment manager,

     `System administration` → **`Deployment manager`**
2. Expand the `Java and process management` tree and click **`Process definition`**.
3. Click **`Java Virtual Machine.`**
4. Update the `Maximum Heap Size` to allow for the increase in heap usage.

However, ensure you do not allocate too much memory to the JVM heap as you run the risk of either limiting native memory for the Java process or causing the system to page memory. Paging is very bad for performance.

You could also consider disabling the HA Manager processes if you do not utilize the services it provides. For more information, see "Determine if you need to use the HA Manager" on page 290.

8.5.5 Too much load due to HA Manager

If you are seeing the following message:

```
HMGR0152W - CPU Starvation detected. Current thread scheduling delay is
{0} seconds.
```

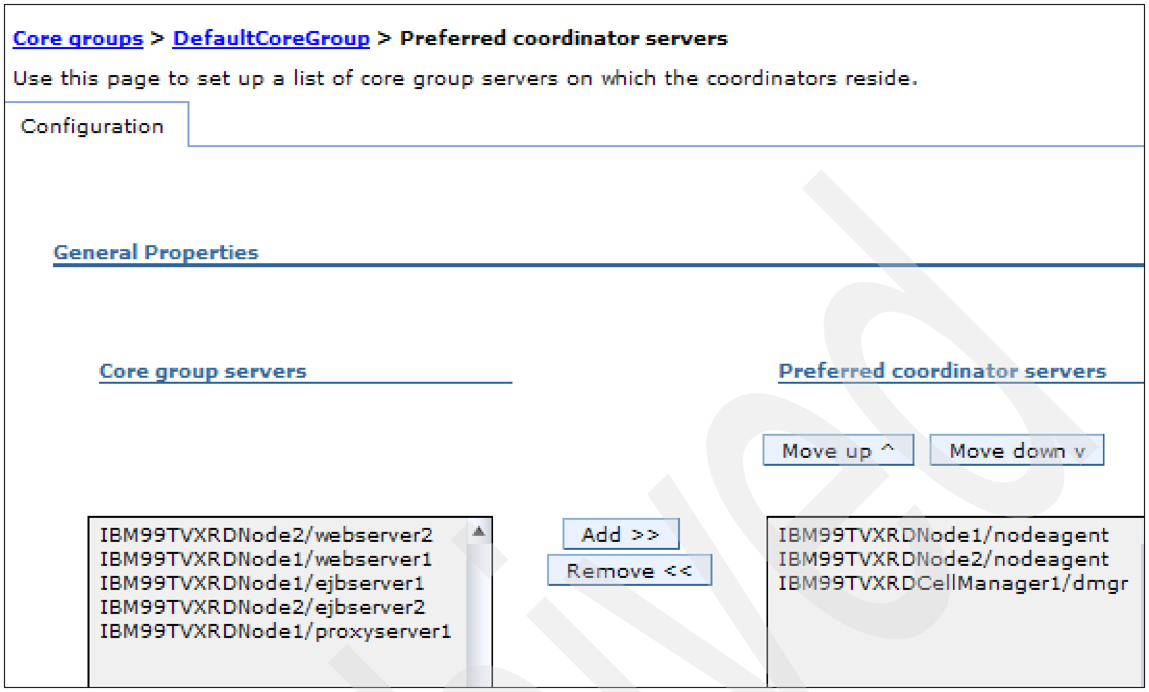
This means that there is a resource constraint on the server where the message appears. Examples of this include JVM heap memory exhaustion, CPU utilization or system memory being paged.

Resolve the problem

Check the CPU usage and physical memory usage stats on the server. Physical memory paging can restrict CPU availability to the HA modules.

If the HA Manager is running on the server experiencing the problem, consider moving it to a process running on a server with more capacity. You can control the servers which can run the core group coordinator by setting the preferred coordinator servers list as shown in Figure 8-9.

1. Navigate to **`Servers`** → **`Core groups`** → **`Core group settings`** → ***`core_group_name`*** → **`preferred coordinator servers`**.


Core groups > DefaultCoreGroup > Preferred coordinator servers

Use this page to set up a list of core group servers on which the coordinators reside.

Core group servers:

```
IBM99TVXRDNode2/webserver2
IBM99TVXRDNode1/webserver1
IBM99TVXRDNode1/ejbserver1
IBM99TVXRDNode2/ejbserver2
IBM99TVXRDNode1/proxyserver1
```

Preferred coordinator servers:

```
IBM99TVXRDNode1/nodeagent
IBM99TVXRDNode2/nodeagent
IBM99TVXRDCellManager1/dmgr
```

Figure 8-9 Preferred coordinator server lists

2. Specify servers that are not often stopped and restarted and that run on hosts with spare CPU and memory resources.

You could also consider disabling the HA Manager processes if you do not utilize the services it provides.

See the following article in the WebSphere Information Center:

- *When to use a high availability manager*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_ha_ham_required.html

8.5.6 Validate the solution

After making the changes described in this section, you will need to synchronize the changes and restart the services. Then recreate the issue and ensure the problem no longer occurs.

If you are still experiencing issues with too much load on the server hosting the HA Manager, go to “The next step” on page 310.

8.6 HA Manager messages in the logs

In many cases, HA Manager problems are only identified by HA Manager related messages appearing in the application server logs rather than by any failure condition. A good system administrator will note these messages and take steps to resolve the underlying problems before a failure condition occurs.

8.6.1 Collect diagnostics

You will see HA MAnager related messages in the application server logs.

- JVM SystemOut and SystemErr logs

8.6.2 Analyze diagnostics

Look for the following in the JVM logs:

- `HMGR0140E - An event indicating that the core group membership is inconsistent was received. Recovery failed. The exception is {0}.`

  If you see this, go to “Inconsistent group membership” on page 293.
- `HMGR1001W - An attempt to receive a message of type {0} for Agent {1} in AgentClass {2} failed. The exception is {3}`

  If you see this, go to “Thread pool problem” on page 299.
- DCSV8050I messages that report inconsistent view sizes across cluster members

  If you see this, go to “Network issues causing split views” on page 299.
- In addition, look for the following message in the logs:

  `HMGR0152W - CPU Starvation detected. Current thread scheduling delay is {0} seconds.`

  This indicates a resource problem on the server. Go to “Too much load due to HA Manager” on page 307.
- `CWRLS0030W: Waiting for HAManager to activate recovery`

  Go to “Unable to obtain lock on transaction log” on page 303.

8.7 The next step

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, there are other things that can go wrong with the HA Manager.

8.7.1 Search online support

If you are sure the problem is with the HA Manager, there are things that you can do before contacting IBM support. First, you should review the documentation that you have gathered for errors that were not addressed in this paper and search support sites for information or fixes. Look for current information available from IBM support on known issues and resolutions on the following IBM support page using the following argument:

`http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCZM52`

Look also at the WebSphere Information Center documentation and technotes for additional resources for diagnosing and fixing HA Manager issues:

- *High availability manager*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_ha_hamanager.html`
- *Core group View Synchrony Protocol*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_ha_netcomp.html`
- *When to use a high availability manager*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_ha_ham_required.html`
- *Tune High Availability (HA) Manager configuration for large cell environments*

  `http://www.ibm.com/support/docview.wss?rs=180&uid=swg21251873`

8.7.2 Re-evaluate the symptoms

If, after going through this process, you still have an undiagnosed problem, we recommend that you go back to *Approach to Problem Determination in WebSphere Application Server V6* at:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf`

Review the problem classifications to see if there are any other components that might be causing the problem.

Contact IBM

If these steps do not resolve your problem, then gather additional information as specified in the following MustGather document and raise a problem record with IBM. The following URL contains a list of the MustGather documentation for HA Manager problems.

http://www.ibm.com/support/docview.wss?rs=180&uid=swg21201016

9

Collecting diagnostic data

This chapter provides information for collecting diagnostic data useful in diagnosing workload management problems.

9.1 Collecting JVM logs

JVM logs, often referred to as SystemOut and SystemErr logs, are created for every WebSphere Application Server process (application server, cluster member, node agent, and deployment manager). They can be found in the following locations:

- WebSphere Application Server V6.x (distributed and i5/OS)

  The JVM log files are by default named SystemOut.log and SystemErr.log. The default location for the SystemOut and SystemErr logs is:

  – *profile_root*/logs/*server_name*/SystemOut.log

  – *profile_root*/logs/*server_name*/SystemErr.log

  The location of application server logs is configurable.

  a. Select **Troubleshooting** → **Logs and Trace** in the navigation bar.

  b. Click the server name.

  c. Select **JVM logs**.

  A page on the administrative console shows the location of the log file.

9.2 Enabling the WLM trace

You can enable and gather WLM trace for an EJB client or the application servers. The following sections will provide guidance on collecting the appropriate traces for WLM problems.

For more information about tracing, see the following WebSphere Information Center articles:

- *Enabling trace on client and standalone applications*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/ttrb_entrstandal.html

- *Object request broker troubleshooting tips*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rtrb_orbcomp2.html

9.2.1 Enabling and gathering trace from a thin application client

To enable and gather trace for a thin application client, do the following:

1. Create a trace properties file by copying the *app_server_root*\properties\TraceSettings.properties file to the same directory as your client application Java archive (JAR) file.
2. Edit the properties file and change the `traceFileName` value to the location for the trace data output. For example, `traceFileName=c:\\temp\\myAppClient.trc`.
3. Edit the properties file to remove `com.ibm.ejs.ras.*=all=enabled` and add:

   ```
   WLM*=all:ORBRas=all
   ```
4. Add the following options to the Java command line that is used to run the client:

   ```
   -DtraceSettingsFile=trace_properties_file
   -Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
   -Djava.util.logging.configureByServer=true
   ```

   Where *trace_properties_file* represents the name of the properties file that you created in the previous steps.

9.2.2 Enabling the trace from a J2EE application using launchClient

Use the following arguments in the *app_server_root*/bin/launchclient.sh or .bat file to collect a J2EE client trace:

- -JVMOptions="-Dcom.ibm.CORBA.Debug=true -Dcom.ibm.CORBA.CommTrace=true"
- -CCtrace=WLM*=all:ORBRas=all
- -CCtracefile=orbtrace.txt
- -CCtraceMode=basic

For example:

```
app_server_root/bin/launchClient.sh ear_file
-JVMOptions="-Dcom.ibm.CORBA.Debug=true -Dcom.ibm.CORBA.CommTrace=true"
-CCtrace=ORBRas=all=enabled -CCtracefile=orbtrace.txt
-CCtraceMode=basic
```

The ORB trace output is captured in a unique trace file named orbtrace.txt in the current directory.

9.2.3 Enabling the trace from the administrative console

To enable and gather the WLM trace for the application servers and WLM clients that are WebSphere application servers, do the following:

1. In the administrative console, expand **Servers** → **Application Servers** → *server_name*.
2. Select **Diagnostic Trace Service**.
3. Set **Maximum File Size** to `200` MB and select the **File radio** button in the General properties section. Set the appropriate number of historical trace files.
4. Navigate to **Change Log Detail Levels** in the Additional properties section.
5. Clear the current trace string and add the following trace string to the General properties field:

   ```
   WLM*=all:ORBRas=all
   ```

   For multiple core groups and one or more core group bridges:

   ```
   WLM*=all:ORBRas=all:Core_Group_Bridge=all
   ```

6. Apply and save your changes.

By default, the trace is logged to a file called `trace.log` in the same location as SystemOut.log and SystemErr.log, *profile_root*/logs/*server_name.*

Additional information for collecting the ORB trace

In addition to the ORB tracing above, you can enable tracing of the ORB GIOP messages. This tracing is referred to by WebSphere Application Server support as a *comm trace*, and is different from the general purpose trace facility. However, you must also have enabled ORB tracing as above for comm tracing to generate any output.

1. In the administrative console, go to the ORB Service page for the application server you want to trace:

   Servers → **Application servers** → ***server_name*** → **Container services** → **ORB service**

2. Check the box for **ORB tracing**.
3. Click **OK**, and then click **Save** to save your settings.
4. Restart the server for the new settings to take effect.

Refer to the following Information Center article for more information:

- *Enabling tracing for the Object Request Broker component*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/rtrb_orbcomp2.html

Disabling the trace

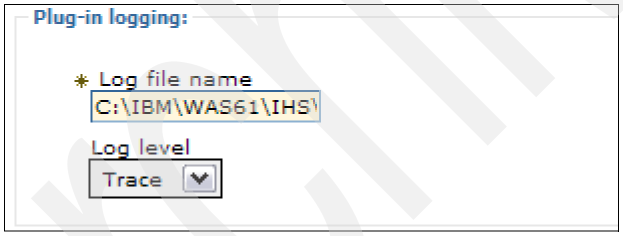
To restore trace state back to normal, use this same process and clear the trace string. Restart the server.

9.3 Collecting the plug-in log

By default the `http-plugin.log` will be in the logs directory under the plug-in installation root:

`plugin_root``/Plugins/logs/`*`web_server_name`*`/http-plugin.log`

You can determine the name and location of your plug-in log by looking in the administrative console for your Web server definition as shown in Figure 9-1. Navigate to **`Web servers`** → ***`web_server_name`*** → **`Plug-in properties`** and check the `Log file name` field.


Plug-in logging:

* Log file name
C:\IBM\WAS61\IHS\

Log level
Trace

Figure 9-1 http-plugin.log location and log level

9.3.1 Setting the log level

The default log level for the plug-in is `Error`, this will only report error conditions such as a server being marked down.

There are several levels of logging at the plug-in that are relevant to monitoring load balancing. The logging levels are cumulative so **`Stats`** includes and builds on the information you get from **`Warn`**, and so on. The levels are:

- **`Warn`** - this log level will report warnings issued by the plug-in.
- **`Stats`** - provides basic statistics on how many requests are sent to each cluster member. It also reports on requests sent to a server to maintain

session affinity. However it does not distinguish between the requested URLs so there is no way to analyze these statistics based on URI or application.

- `Detail` - provides detailed statistics on how each request was processed and how the plug-in chose to send a given request to a particular server.
- `Debug` - similar to detail but with more information reported.
- `Trace` - provides the highest level of detail about request processing. A plug-in trace will get very large very quickly.

The logging level you should choose will depend on the complexity of your load balancing environment. If you have only one application being load balanced, then `Stats` will be sufficient to determine the load distribution. However if you have multiple applications, some with session affinity and some without, then you will need to move to a higher log level. You will probably be trying to resolve a production problem. Given this, it may be best to collect `Trace` data so that you do not have to keep running tests and traces at subsequently higher log levels to collect more information in your production environment.

Tip: If you have not enabled automatic generation and propagation of the plug-in, you will need to do this manually for the change to log level to take effect.

You do not need to restart your Web server for this change to take effect. By default, the plug-in will reload its configuration every 60 seconds. You only need wait for the reload interval to pass.

Part 6

JMS problem determination

This part contains the following IBM Redpaper:

- *WebSphere Application Server V6.1: JMS Problem Determination*, REDP-4330 by Richard Coppen, Gareth Bottomley, Brian De Pradine, Sarah Drewery, Graham Hopkins, Philip Nickoll, Alasdair Nottingham, Matthew Roberts, Dave Vines, David Ware, and Bryan Williams

10

Introduction to JMS application problem determination

This chapter describes the first steps you take to diagnose a JMS related problem in a WebSphere Application Server V6.1 environment. It helps you determine which message provider is in use, how to find and collect relevant information, and how to find problem determination information in the rest of this IBM Redpaper that is applicable to your particular situation.

10.1 Identify your messaging provider type

This publication addresses problems that can occur when using two of the messaging providers supported by WebSphere Application Server. To diagnose a problem, you must first determine which provider you are using.

10.1.1 Default messaging provider

The default messaging provider is a component of the WebSphere Application Server Version 6.1 that is installed by default. You can configure it to support JMS messaging as part of a service integration bus.

For a detailed description of WebSphere default messaging concepts and architecture, refer to Part 2 of *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

10.1.2 WebSphere MQ as the JMS provider

Alternatively, you can configure WebSphere Application Server Version 6.1 to use WebSphere MQ as a messaging (JMS) provider.

For more information about WebSphere MQ as a JMS provider, see *Configuring JMS resources for the WebSphere MQ messaging provider* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tmj_admrm.html

10.1.3 Default messaging provider interoperation with WebSphere MQ

You can also configure WebSphere Application Server default messaging to interoperate with your existing WebSphere MQ infrastructure.

There are various forms of interoperation, and understanding the difference is important.

For help with understanding the interoperation choices, see:

- *Managing WebSphere Application Server Version 5 JMS use of WebSphere Application Server Version 6 messaging resources* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.express.doc/tasks/tjq0000_.html

- *Interoperating with WebSphere MQ using a WebSphere MQ link* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.zseries.doc/tasks/tjc9999_.html
- *Interoperating with WebSphere MQ on z/OS using WebSphere MQ server* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.zseries.doc/tasks/tjfp0031_.html
- *Learning about messaging with WebSphere Application Server* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tm_learn.html

10.2 Default messaging provider problem determination

The *service integration bus* is responsible for asynchronously transferring messages from message producing applications to queues and topic spaces. *Message points* within the messaging engines are the physical location of the destination. From the message points, the messages are transferred to the consuming application.

The first step in problem determination is to understand the bus and messaging engine topology that are used in your environment. For example, your topology might be:

- One service integration bus and one messaging engine.
- One service integration bus with clustering of messaging engines. Determine if the cluster is configured for high availability or workload management.
- One service integration bus with multiple messaging engines.
- One service integration bus with a foreign bus.

The type of topology that is used might be a decision point during the problem determination process. Use this information to identify the messaging engines involved in your application flow.

10.2.1 Identify your messaging engine topology

Messages are held on message points associated with the messaging engine. A message point can be a:

- Queue point
- Publication point
- Mediation point

When there is only one messaging engine within a service integration bus, both the producer and consumer applications can be connected to that single messaging engine only where the message point of the queue or topic space is located. This simple case is illustrated (for point-to-point messaging) in Figure 10-1.

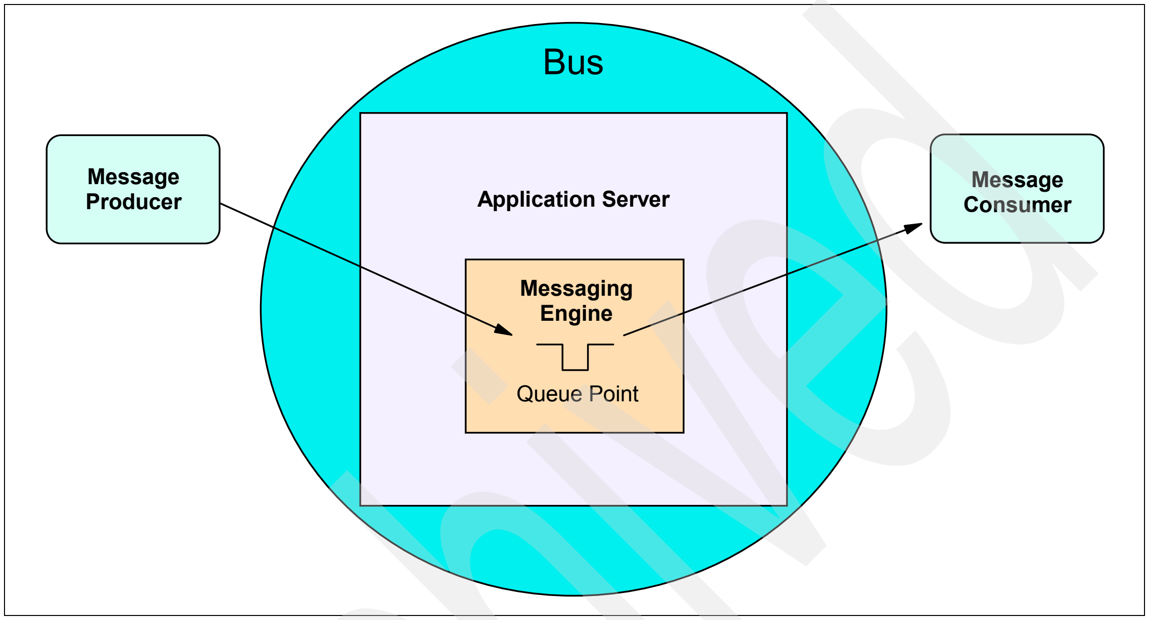


Figure 10-1 Single messaging engine topology

When using the default messaging provider in a distributed environment, you can configure multiple bus members and messaging engines for a single bus.

In addition, you can configure clusters as members of the bus, each with a scope for multiple messaging engines. In a *highly available (HA) configuration,* one messaging engine is active on one of the application servers in the cluster at any given time. If the messaging engine or application server fails, the messaging engine is started on another application server in the cluster. In a *workload management (WLM) configuration,* each application server in the cluster has an active messaging engine.

This more complex case is illustrated in Figure 10-1.

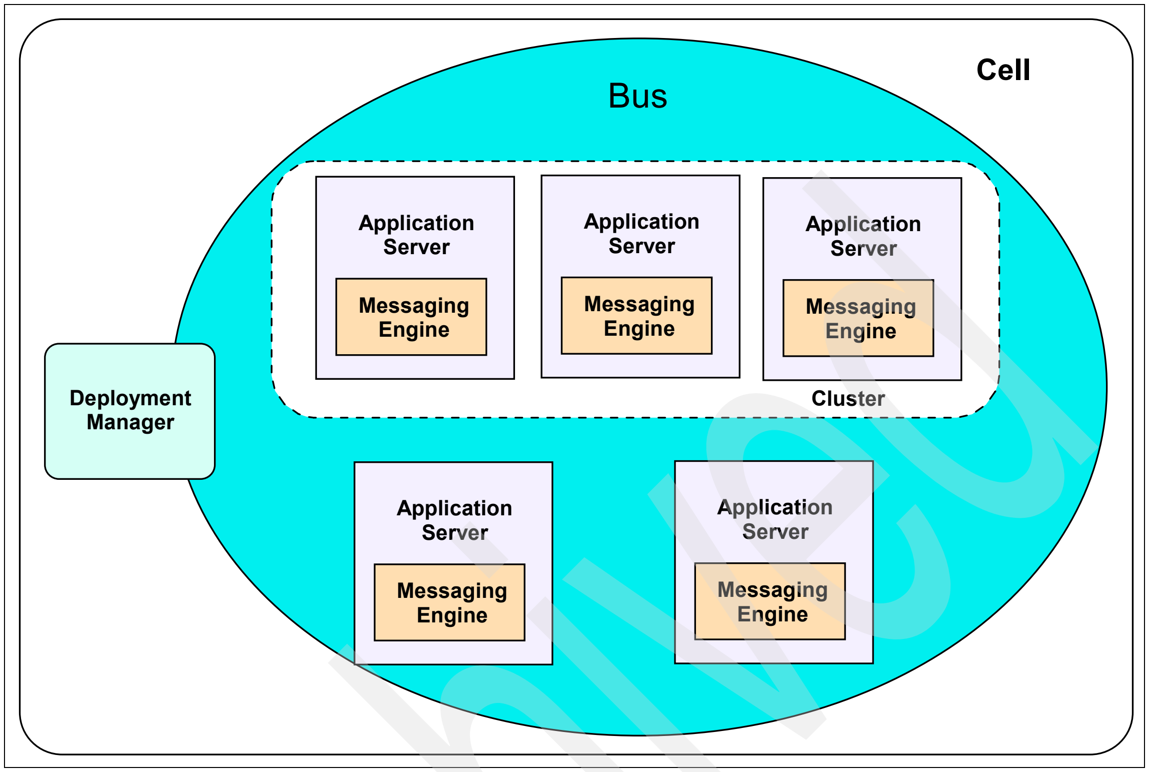


Figure 10-2 Multiple messaging engine topology

If your topology includes multiple messaging engines in the same bus, the producing and consuming applications can be connected to different messaging engines. And, in the case of point-to-point messaging, the producing and consuming applications can be connected to a messaging engine that is remote to the location of the actual message point.

This topology is illustrated in Figure 10-3 on page 326.

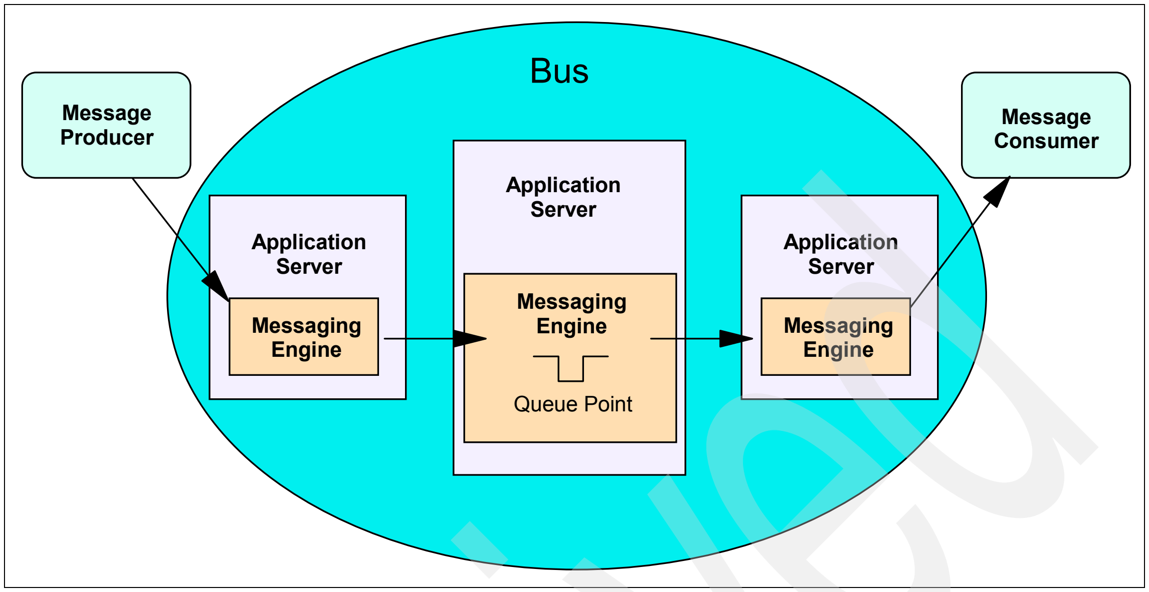


Figure 10-3 Message producer and message consumer connected to different messaging engines

You can configure a bus to connect to and exchange messages with other messaging infrastructures. You define the remote messaging infrastructure to the bus as a *foreign bus*, which can be another service integration bus in the same cell or in a different cell. A foreign bus can also be a WebSphere MQ infrastructure, as shown in Figure 10-4 on page 327.

The foreign bus represents the remote bus type. To identify a specific bus, you need either a WebSphere MQ link or service integration bus link.

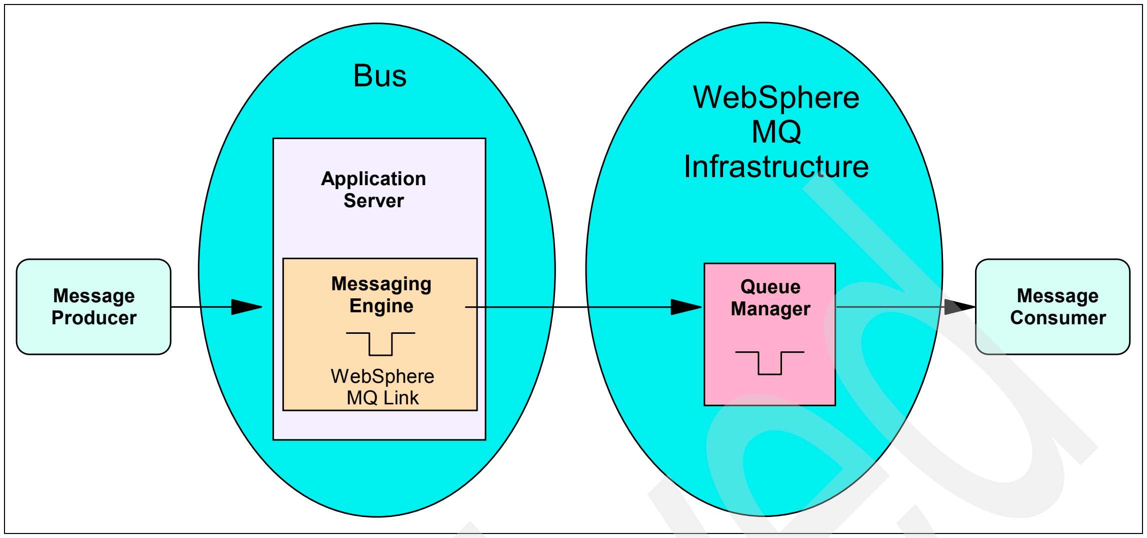


Figure 10-4 WebSphere MQ as a foreign bus

10.2.2 Determine the messaging engine connected to an application

If your application uses a JMS JNDI connection factory, inspect its properties to identify where it will connect.

In the WebSphere administrative console:

1. Select **Resources** → **JMS**.
2. Select the type of connection factory you are using.
3. Set the appropriate scope.
4. Click on the connection factory name to open the details page.
5. The connection properties display in the "Connection" area.
6. If a bus member or messaging engine name is defined in the Target field, you can use this to narrow the possible messaging engines to which the application can connect.

   Unless the Target significance field is set to `Required`, the messaging engine specified might not necessarily become the configured target. Any messaging engine in the bus can be used.

Alternatively, you can use one of these methods to identify the exact messaging engine to which the application connects at run time:

- Call the `toString()` method of the connection object. This method includes a reference to the connected messaging engine.

- Enable the SIBJms_External trace for a JMS application and inspect the output for a reference to the connected messaging engine.

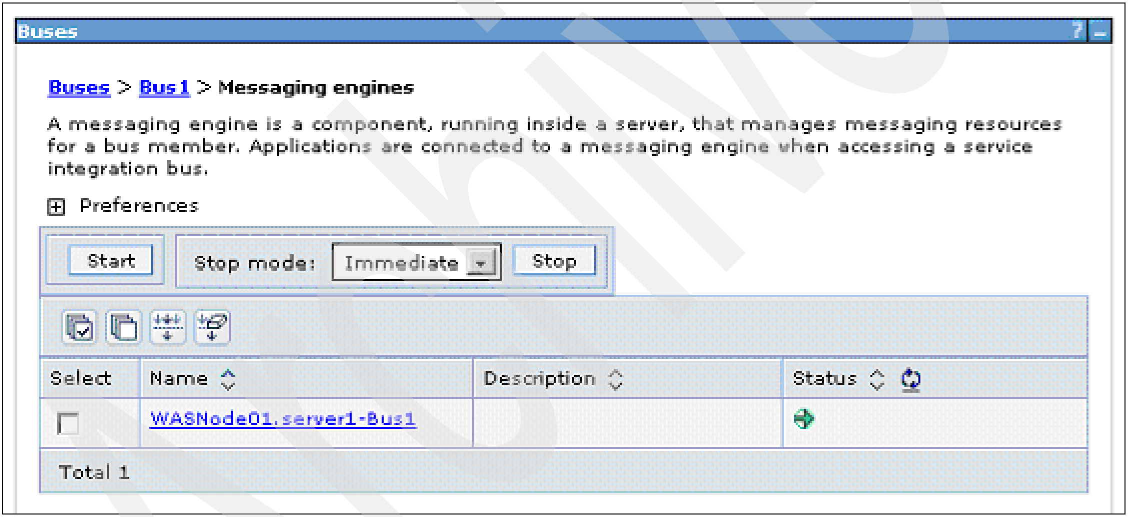
10.2.3 Determine your messaging engine status

If you suspect that you have a default messaging problem, first make sure that the appropriate messaging engines are running.

The simplest method to check the status of the messaging engines is to use the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.

You see a list of messaging engines and their status, as illustrated in Figure 10-5.



A messaging engine is a component, running inside a server, that manages messaging resources for a bus member. Applications are connected to a messaging engine when accessing a service integration bus.

Stop mode: Immediate

Select	Name	Description	Status
☐	WASNode01.server1-Bus1		➔

Figure 10-5 Messaging engine status

One of four messaging engine states displays in the Status column of this administrative console's messaging engine panel:

- A light green arrow indicates that the messaging engine has entered the *Transitive* starting state.
- A dark green arrow indicates that the messaging engine is in *Started* state.
- A red cross indicates that the messaging engine is in *Stopped* state.

- A grey circle with a line through it indicates that the messaging engine is in *Unavailable* state.

You can start or stop the messaging engine by checking the box to the left of the messaging engine and clicking the appropriate button.

If you are stopping the messaging engine, select a stop mode first. An *immediate* stop stops the messaging engine when all messaging operations that are being carried out at the time of the stop request are complete. A *force* stop stops the messaging engine without allowing messaging operations to complete and applications are forcibly disconnected.

10.2.4 Finding queued messages

To track messages that might be held in message points, you need to understand your topology. In a simple topology with one messaging engine, looking for messages is simply a matter of viewing the message points on the messaging engine.

If your topology includes multiple messaging engines in the same bus, the producing and consuming applications can be connected to different messaging engines. And, in the case of point-to-point messaging, the producing and consuming applications can be connected to a messaging engine that is remote to the location of the actual message point. Finding messages can involve tracking messages across messaging engines by looking at the remote message points. In Version 6.1, if you have multiple buses in the topology, messages that are queued for a remote bus cannot be viewed on message points.

To find queued messages, you first need to locate all of the message points that might contain messages and then check them for messages.

The following steps show how to check for messages in a point-to-point messaging context. However, you can use similar steps in a publish/subscribe environment; simply substitute references to *queue* and *queue point* with *topic space* and *subscription*, respectively.

To check for messages:

1. If the producing application is a JMS application, use the JMS JNDI destination definition to find the service integration bus destination to which it maps. Use the administrative console to:
   a. Select **Resources** → **JMS**.
   b. Select the destination type of **queue**.
   c. Click on the queue name to open the details page.

   d. The service integration bus destination will be in the **Queue name** field in the Connection area.

2. Determine if the service integration bus destination is an alias:

   a. Select **Service integration** → **Buses**.

   b. Click the bus name to open the details page.

   c. Click **Destinations.**

      The type column indicates if the destination is an alias destination.

      If this is the case:

      i. Click on the destination name.

      ii. The service integration bus destination is referenced under the General properties heading as the **Target identifier**.

      Use the target destination, and repeat this step until a queue destination is reached.

3. Check all queue points of the queue destination for messages.

   If multiple queue points exist for a given queue, the queue is owned by a clustered messaging engine. In such cases, perform these steps on each of the queue points:

   a. Check each queue point for messages.

   b. Where messages are found, click on the message to display its details. If the message is not one you are looking for, continue to the next step.

4. If the queue is mediated, repeat step 3 for the mediation points associated with the queue. If messages are queued for mediation, see Chapter 26, "Mediation problem determination" on page 565.

If you cannot locate your message, check that no applications are consuming messages from the message points and, therefore, removing the messages before they can be viewed in the administrative console.

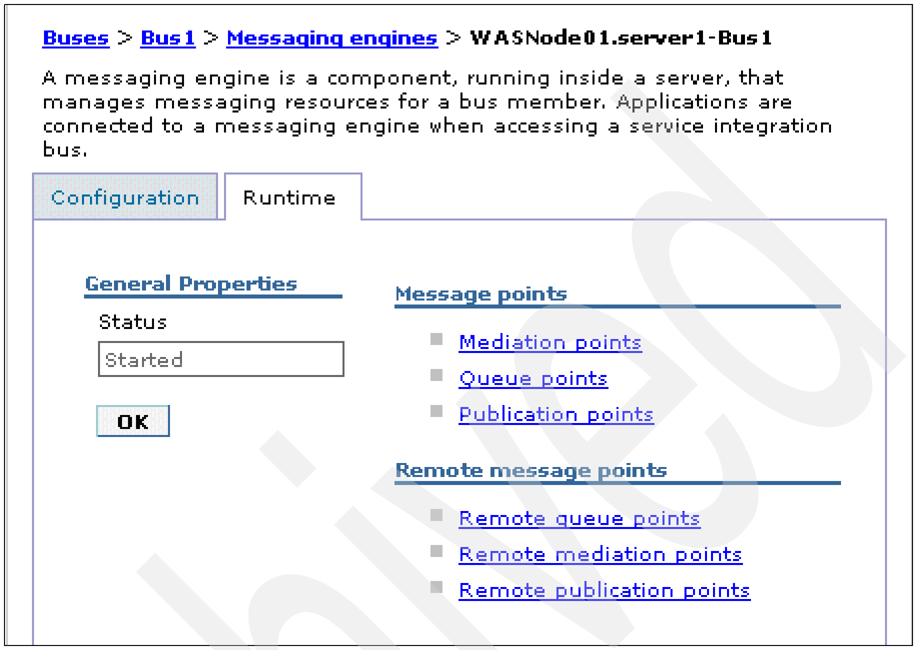
The following sections explain how messages flow across message points and how to look at the message points involved in the flow.

Message points

When a message is held on a message point, you can view it using the administrative console, as shown in Figure 10-6 on page 331:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.

4. Click the messaging engine name to open the details page.
5. Select the **Runtime** tab.

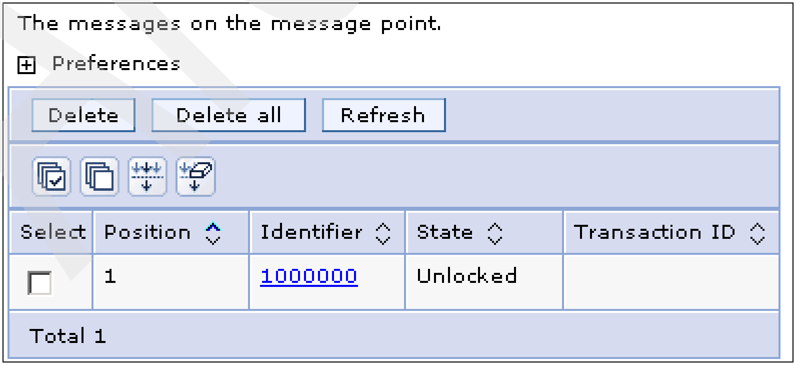


Buses > Bus1 > Messaging engines > WASNode01.server1-Bus1

A messaging engine is a component, running inside a server, that manages messaging resources for a bus member. Applications are connected to a messaging engine when accessing a service integration bus.

Figure 10-6   Messaging engine runtime information

6. To view messages held on a message point (as illustrated in Figure 10-7), select the message point type, for example, **Queue points**.
7. Click on the message point name.
8. Click **Messages**.



The messages on the message point.

Select	Position	Identifier	State	Transaction ID
☐	1	1000000	Unlocked	
Total 1				

Figure 10-7   Messages held on a message point

The value in the Identifier column does not uniquely identify the message within the service integration bus and has no relation to the JMS message identifier supplied to the application. However, if you click on the identifier, you see the message details, including the API Message properties. These properties contain the identifier that can be obtained by the application from the JMS Message object and the system message ID that uniquely identifies the message within the service integration bus.

Remote message points

The mechanisms that are used by the service integration bus to asynchronously transfer messages between messaging engines in a manner that maintains the high levels of reliability and recoverability (dependent on the message's chosen quality of service) are known as *remote message points*. These are runtime objects, created dynamically by each messaging engine.

There are three types of remote message points, one for each of the corresponding message point types: queue, publication, and mediation (see Figure 10-6 on page 331).

A remote message point on a messaging engine identifies the host messaging engine and its message point. The remote message point manages the state of messages flowing to and from the message point. You can think of it as a proxy to the real message point on a remote messaging engine.

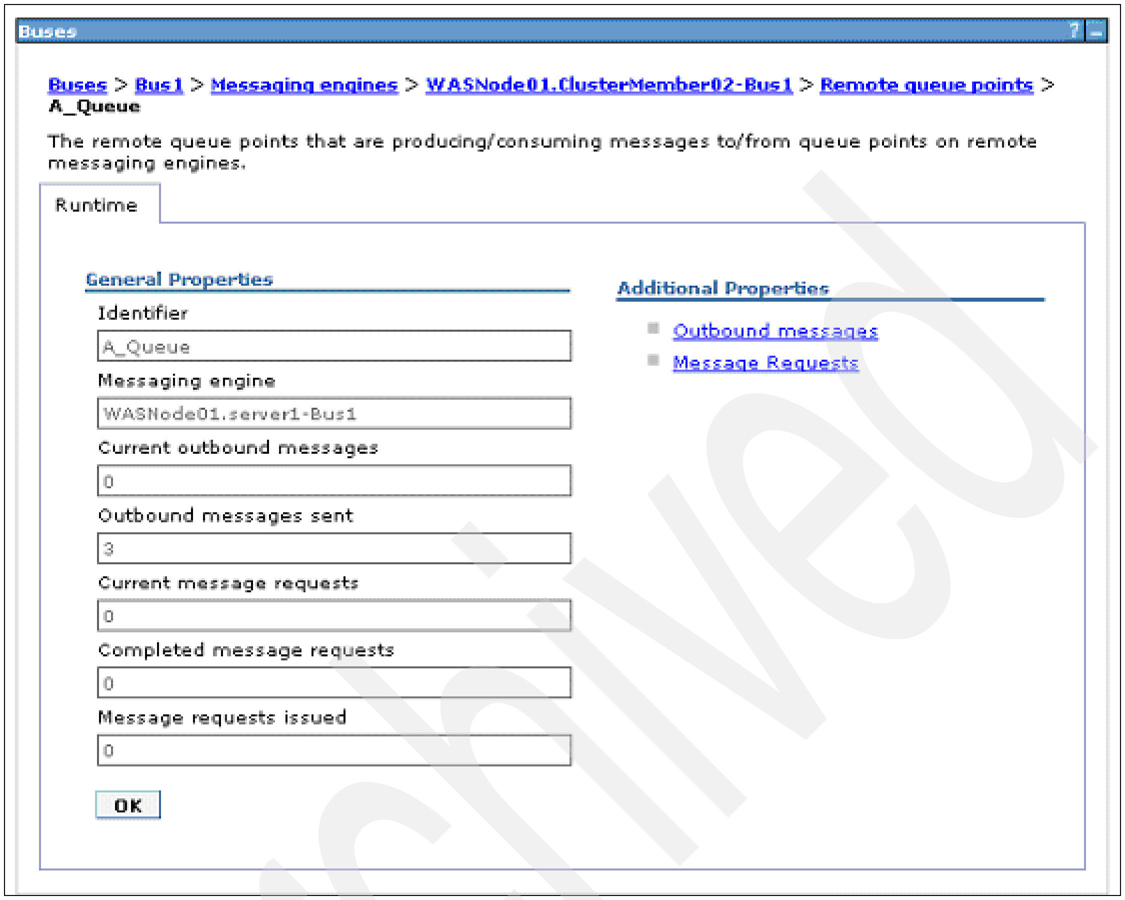
Finding messages queued for transfer

Each remote message point maintains a queue of messages that are currently waiting to be transferred to the remote messaging engine that hosts the message point. It has information about the current state of those messages being sent or received (to be consumed by local consuming applications).

To view this information, use the administrative console to:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.
4. Click the messaging engine name to open the details page.
5. Select the **Runtime** tab.
6. Click on the type of remote message point.
7. Click on the remote message point name.

Figure 10-8 on page 333 illustrates this for A_Queue, a queue-type message point.


Buses

Buses > Bus1 > Messaging engines > WASNode01.ClusterMember02-Bus1 > Remote queue points > A_Queue

The remote queue points that are producing/consuming messages to/from queue points on remote messaging engines.

Runtime

General Properties

Identifier: A_Queue

Messaging engine: WASNode01.server1-Bus1

Current outbound messages: 0

Outbound messages sent: 3

Current message requests: 0

Completed message requests: 0

Message requests issued: 0

Additional Properties

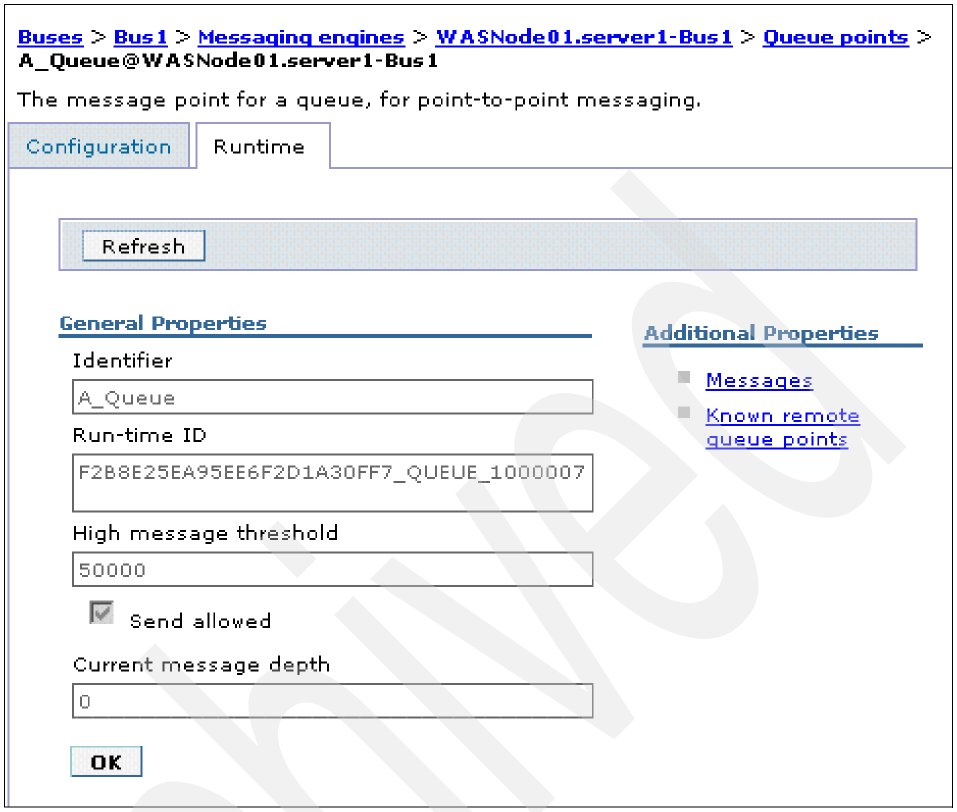
- Outbound messages
- Message Requests

Figure 10-8 Remote queue point

Known remote message points

Each active remote message point has a corresponding partner on the remote messaging engine with which it is communicating. These partner objects are referred to as *known remote message points*. You can view them on the messaging engine that owns the message point; on the **Runtime** tab for the message point, look below **Additional Properties**.

Figure 10-9 on page 334 shows known remote queue points for A_Queue, a queue-type message point.


Buses > Bus1 > Messaging engines > WASNode01.server1-Bus1 > Queue points > A_Queue@WASNode01.server1-Bus1

The message point for a queue, for point-to-point messaging.

General Properties

Identifier: A_Queue

Run-time ID: F2B8E25EA95EE6F2D1A30FF7_QUEUE_1000007

High message threshold: 50000

[✓] Send allowed

Current message depth: 0

Figure 10-9 Known remote queue points

Point-to-point messaging queue points

Figure 10-10 on page 335 shows the remote queue points and known remote queue points listed for three messaging engines (ME).

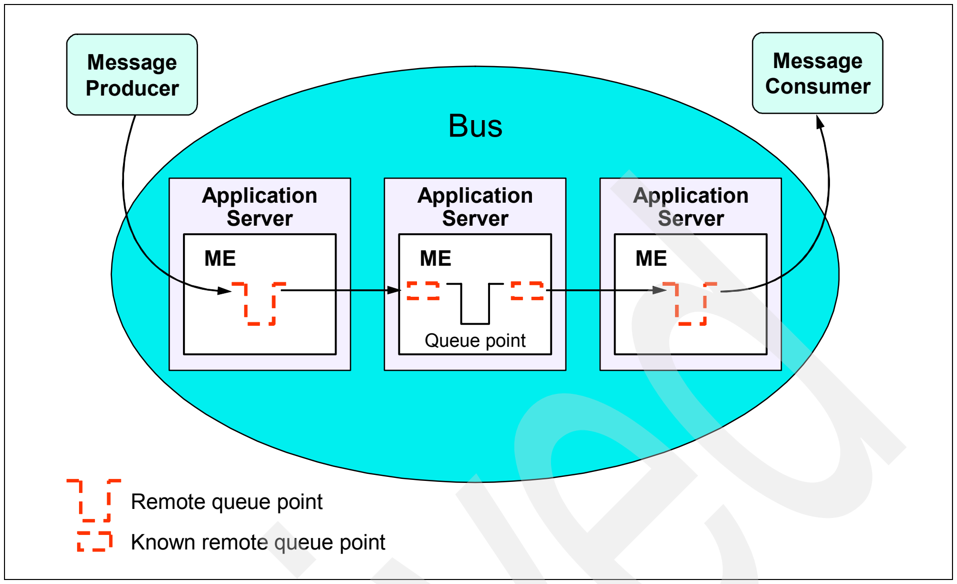
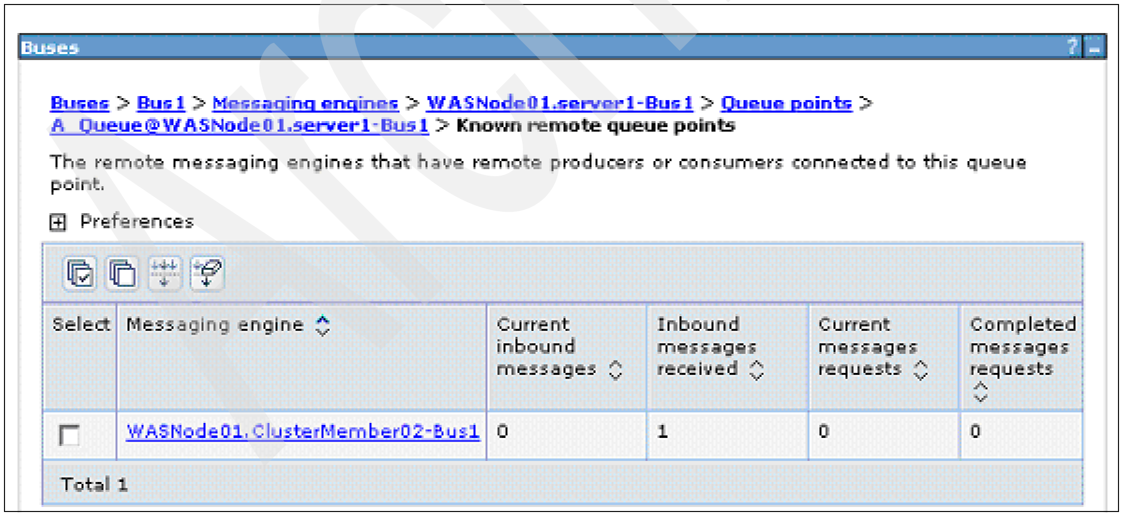


Figure 10-10 Remote queues in point-to-point messaging

The known remote queue points maintain information (Figure 10-11) on which messages have been received from the remote queue point and any current, in-flight messages about to be delivered. They also maintain a history of messages that are currently being consumed on the remote messaging engine.



Buses > Bus1 > Messaging engines > WASNode01.server1-Bus1 > Queue points > A_Queue@WASNode01.server1-Bus1 > **Known remote queue points**

The remote messaging engines that have remote producers or consumers connected to this queue point.

Select	Messaging engine	Current inbound messages	Inbound messages received	Current messages requests	Completed messages requests
☐	WASNode01.ClusterMember02-Bus1	0	1	0	0
Total 1					

Figure 10-11 Known remote queue point information for example

In a healthy running system, the state of a remote queue point and its partner known queue message point should be in-step and probably empty (or very nearly empty) of queued messages. However, the introduction of multiple places where messages might be queued obviously increases the potential number of places where messages can be held up on the way to ensure the reliable asynchronous delivery of messages. Although these messages are not lost, they can become stuck due to some external influence. Therefore, it is sometimes necessary to inspect this runtime information to either diagnose problems in message delivery or to simply monitor the current state of the system.

Publish/subscribe messaging queue points

Similar to the remote queue points used to transfer messages from one messaging engine to another for point-to-point messages, *remote publication points* are used to transfer messages from one messaging engine to another for publish/subscribe messaging.

Each messaging engine in a bus has a *publication point* that represents a topic space. Any application that produces messages for a topic space produces them to the publication point on the connected messaging engine. This message is then forwarded to each subscription that is interested in this message. If any of those subscriptions are located on another messaging engine in the bus, a remote publication point is used to transfer those messages to the remote messaging engine's publication point and, from there, it is forwarded to the subscription.

If subscriptions exist on more than one remote messaging engine, the message is forwarded to multiple remote publication points, as illustrated in Figure 10-12 on page 337.

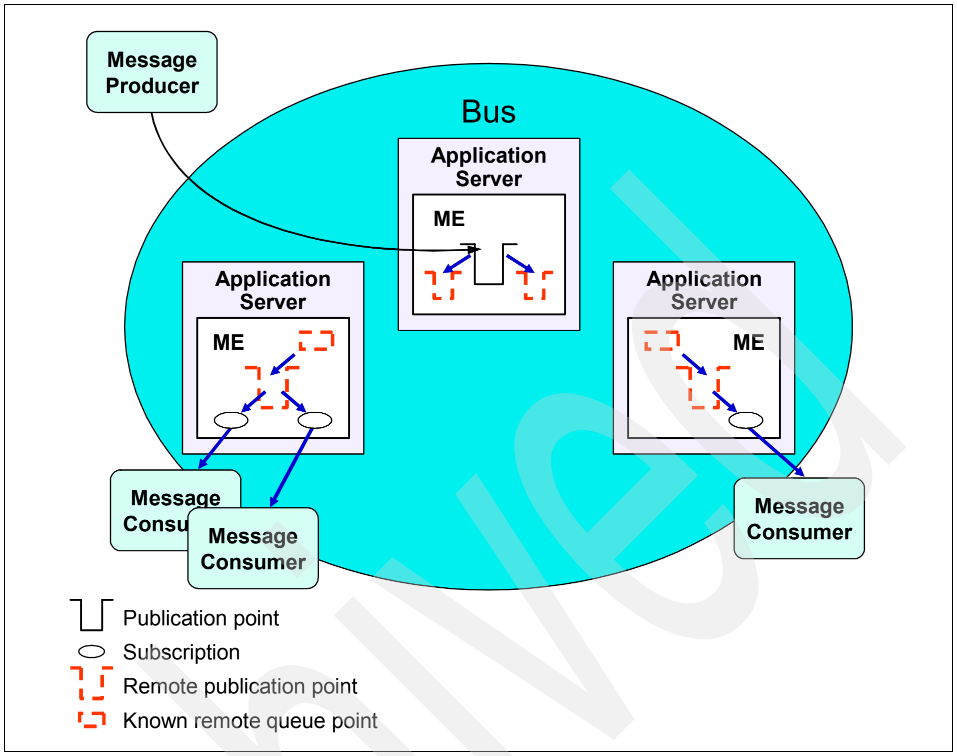


Figure 10-12 Publish/subscribe topology

For further information about messaging between messaging engines in a Network Deployment environment, see *Remote message points* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjo_remote_msg_pts.html

10.2.5 Finding lost messages

If messages are not being consumed by the application and they are not queued where you expect them, you can do several things to discover the reason.

If you have multiple buses, messages might not be viewable in the administrative console and will, therefore, be very hard to track down. Contact IBM technical support for assistance.

Prevent applications from consuming messages on a queue

First, identify all possible applications consuming messages from the queue, including aliases to the queue. If you cannot identify all possible applications, you can disallow all consumers on the queue to prevent applications from consuming messages by setting the **Receive allowed** property of the queue destination on the bus.

To disallow consumers on the queue:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations**.
4. Click on the destination name where you wish to disallow consumers.
5. Clear the **Receive allowed** check box.

You might need to resend messages and repeat the above steps to check for queued messages. If you are still unable to locate your messages, proceed to the next step.

Check the message reliability

If messages have a reliability of *best effort nonpersistent,* the messages might have been discarded due to system resource constraints. This is one drawback of using best effort nonpersistent messages. Consider changing the reliability to *express nonpersistent* and resending the messages.

More information about message reliability can be found in *Message reliability levels* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjj9000_.html

Check for expiration time

You should check whether the sending application sets an expiration time in the message. If this time has elapsed, then the message will have been removed from the queue.

Check the exception destination

The message might have been moved to an exception destination. Check the configuration of the queue to determine its exception destination policy:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations**.
4. Click on the destination name.

The exception destination policy is specified under the Exception destination heading.

Finally, if a specific destination has been identified, check that destination for the message. Where the system exception destination is being used, check the system exception destination for each messaging engine that hosts a queue point. If the message is found on one of the above destinations, display the message details to find the cause for the exception. Resolve the problem identified.

10.3 Collect diagnostics

After you understand your bus and messaging engine topology, determining the specific problem involves inspecting diagnostic information generated by the applications, application servers, and messaging engines in the topology.

Continue your diagnostic approach by collecting these diagnostic data:

- Application log

  When an application receives an exception from a JMS API call, the full stack trace of this exception, including all linked exceptions can be the most useful starting point for problem determination.

  When an exception is caught by application code, it is the responsibility of that application to produce the full stack trace of that exception. For example, the application can call the `printStackTrace` method on a caught exception to produce a full stack trace to the standard error location.

  The standard error location is as follows:

  – If the application is running in the client container, the stack trace is sent to the standard error of the launchClient console.

  – If the application is running in an application server, the stack trace is sent to the SystemErr log of the application server.

  An application hosted in an application server (such as a message-driven bean (MDB)) might choose not to catch an exception, or to re-throw it to its container. The container then logs the exception to the JVM logs of the application server. It might be necessary to check both the SystemOut and SystemErr JVM logs in this case.

- Application server JVM log files for the messaging engines

  The messaging engines, and other application server components, within your topology log messages to the JVM logs of the application server within which they are running. These messages can describe expected events, such as messaging engines starting and stopping, as well as error conditions.

Collect the log for each server where the applications and messaging engines can run. In a high availability configuration, the messaging engine is active on only one cluster member at a time, but it can be started on any member. In a WLM configuration, there might be multiple messaging engines active simultaneously on multiple cluster members.

If you are collecting diagnostic data to send to IBM technical support, see 27.2, “Contact IBM” on page 582.

10.3.1 Collecting JVM logs

JVM logs, often referred to as application server or SystemOut and SystemErr logs, are created for every WebSphere Application Server process (application server, cluster member, node agent, and deployment manager).

To find them, look in the these locations:

- On WebSphere Application Server V6.*x* for z/OS:

  The JVM logs are located in the address space output. Usually a section labeled SYSOUT contains diagnostic data from the JVM that runs in the servant region.

- On WebSphere Application Server V6.*x* (distributed and i5/OS):

  The JVM log files are, by default, named SystemOut.log and SystemErr.log. The default location for the SystemOut and SystemErr logs is:

  – *profile_root*/logs/*server_name*/SystemOut.log

  – *profile_root*/logs/*server_name*/SystemErr.log

  The location of application server logs is configurable:

  a. Select **Troubleshooting** → **Logs and Trace** in the navigation bar.

  b. Click on the server name.

  c. Select **JVM logs.**

  This page shows the location of the log file.

If the error occurred some time ago, you might need to check older versions of the log. For example:

```
SystemOut_05.06.07_10.28.48.log
```

If the error occurs for an application running in a cluster and you do not know the specific server where the error occurred, you might need to collect SystemOut and SystemErr logs from all the active servers in the cluster.

JVM log settings

For information about JVM log settings that affect file rotation, log location, and other SystemOut and SystemErr log behavior, see this Information Center section, *Java virtual machine (JVM) log settings,* at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/utrb_jvmlogs.html`

Logs in the z/OS environment

For the WebSphere on z/OS architecture an application server consists of a control region (CR) address space, and one or more servant region (SR) address spaces. An application server might also include a third type of address space called the Control Region Adjunct (CRA). The CRA is the address space for any messaging engines that might exist for the application server.

Messaging diagnostic information is located in the Servant Region (SR) and Control Region Adjunct (CRA) job logs.

For detailed information about diagnosing problems on the z/OS platform, see *WebSphere for z/OS Problem Determination Means and Tools*, REDP-6880.

10.4 Guide to chapters

After you have gathered the basic information about your messaging configuration, you can approach problem determination several ways using the information in this IBM Redpaper.

To find the correct information:

1. If your messaging engine will not start, start with Chapter 11, “Messaging engine problem determination” on page 355.
2. If you know the error messages being produced, scan the tables in 10.5, “Guide to chapters by message” on page 342 for your message and see the chapter indicated.
3. If your application is receiving exceptions, scan the table in 10.6, “Guide to chapters by exception” on page 348 and see the chapter indicated.

4. If no error messages are being produced, or you are not sure, scan the list of symptoms addressed in each chapter in “Guide to chapters by symptom” on page 350.
5. If you feel sure your symptoms are not addressed, see Chapter 27, “The next step” on page 581 for information about contacting IBM technical support.

10.5 Guide to chapters by message

If you have already isolated an error message from an application server, and are unsure if you have a messaging problem, use Table 10-1 through Table 10-21 on page 348 as a reference to find problem determination activities described in this IBM Redpaper.

Table 10-1 CWSIA messages

Specific message	Chapter
CWSIA0004E	► Chapter 25, “Default messaging provider security” on page 549
CWSIA0006E	► Chapter 25, “Default messaging provider security” on page 549
CWSIA0059E	► Chapter 15, “JMS application problem determination” on page 401
CWSIA0062E	► Chapter 15, “JMS application problem determination” on page 401 ► Chapter 24, “Foreign bus problem determination” on page 529
CWSIA0063E	► Chapter 15, “JMS application problem determination” on page 401
CWSIA0067E	► Chapter 15, “JMS application problem determination” on page 401
CWSIA0069E	► Chapter 15, “JMS application problem determination” on page 401 ► Chapter 25, “Default messaging provider security” on page 549
CWSIA0085E	► Chapter 15, “JMS application problem determination” on page 401
CWSIA0086E	► Chapter 15, “JMS application problem determination” on page 401
CWSIA0087W	► Chapter 15, “JMS application problem determination” on page 401
CWSIA0090E	► Chapter 15, “JMS application problem determination” on page 401 ► Chapter 25, “Default messaging provider security” on page 549
CWSIA0241E	► Chapter 11, “Messaging engine problem determination” on page 355 ► Chapter 15, “JMS application problem determination” on page 401

Table 10-2 CWSIC messages

Specific message	Chapter
CWSIC3080E	► Chapter 22, “WebSphere MQ link problem determination” on page 499
CWSIC3011E	► Chapter 22, “WebSphere MQ link problem determination” on page 499
CWSIC3096I	► Chapter 22, “WebSphere MQ link problem determination” on page 499
CWSIC3108E	► Chapter 22, “WebSphere MQ link problem determination” on page 499
CWSIC1001E	► Chapter 11, “Messaging engine problem determination” on page 355 ► Chapter 15, “JMS application problem determination” on page 401
CWSIC8002E	► Chapter 20, “WebSphere MQ server problem determination” on page 481

Table 10-3 CWSID messages

Specific message	Chapter
CWSID0027E	► Chapter 11, “Messaging engine problem determination” on page 355 ► Chapter 17, “Clustering problem determination” on page 439
CWSID0035E	► Chapter 17, “Clustering problem determination” on page 439

Table 10-4 CWSII messages

Specific message	Relevant topics
CWSII0205W	► Chapter 25, “Default messaging provider security” on page 549
CWSII0219W	► Chapter 24, “Foreign bus problem determination” on page 529
CWSII0211 through CWSII0240	► Chapter 25, “Default messaging provider security” on page 549
CWSII0242 through CWSII0259	► Chapter 25, “Default messaging provider security” on page 549

Table 10-5 CWSIJ messages

Specific message	Chapter
CWSIJ0063E	► Chapter 11, “Messaging engine problem determination” on page 355 ► Chapter 15, “JMS application problem determination” on page 401

Table 10-6 CWSIK messages

Specific message	Relevant topics
CWSIK0018E	► Chapter 25, “Default messaging provider security” on page 549
CWSIK0025E	► Chapter 15, “JMS application problem determination” on page 401
CWSIK0033E	► Chapter 15, “JMS application problem determination” on page 401
CWSIK0101W	► Chapter 26, “Mediation problem determination” on page 565
CWSIK0102E	► Chapter 26, “Mediation problem determination” on page 565
CWSIK0103E	► Chapter 26, “Mediation problem determination” on page 565
CWSIK0104E	► Chapter 22, “WebSphere MQ link problem determination” on page 499

Table 10-7 CWSIP messages

Specific message	Relevant topics
CWSIP0309E	► Chapter 25, “Default messaging provider security” on page 549
CWSIP0771I	► Chapter 26, “Mediation problem determination” on page 565
CWSIP0773I	► Chapter 26, “Mediation problem determination” on page 565
CWSIP0775I	► Chapter 26, “Mediation problem determination” on page 565
CWSIP0811W	► Chapter 20, “WebSphere MQ server problem determination” on page 481.
CWSIP0815W	► Chapter 20, “WebSphere MQ server problem determination” on page 481.

Table 10-8 CWSIS messages

Specific message	Chapter
CWSIS0002E	► Chapter 11, “Messaging engine problem determination” on page 355 ► Chapter 17, “Clustering problem determination” on page 439
CWSIS0002E with CWSIS*xxxx*E	► Chapter 13, “Message store with data store persistence” on page 369
CWSIS0002E with CWSOM*xxxx*E	► Chapter 14, “File store problem determination” on page 389
CWSIS1002E	► Chapter 14, “File store problem determination” on page 389
CWSIS1501E	► Chapter 13, “Message store with data store persistence” on page 369 ► Chapter 17, “Clustering problem determination” on page 439
CWSIS1514E	► Chapter 13, “Message store with data store persistence” on page 369

Specific message	Chapter
CWSIS1519E	► Chapter 13, “Message store with data store persistence” on page 369 ► Chapter 17, “Clustering problem determination” on page 439
CWSIS1522E	► Chapter 13, “Message store with data store persistence” on page 369
CWSIS1524E	► Chapter 13, “Message store with data store persistence” on page 369 ► Chapter 17, “Clustering problem determination” on page 439
CWSIS1535E	► Chapter 13, “Message store with data store persistence” on page 369 ► Chapter 17, “Clustering problem determination” on page 439
CWSIS1538I	► Chapter 13, “Message store with data store persistence” on page 369
CWSIS1545I	► Chapter 13, “Message store with data store persistence” on page 369
CWSIS1546I	► Chapter 13, “Message store with data store persistence” on page 369
CWSIS1573E through CWSIS1576E	► Chapter 14, “File store problem determination” on page 389

Table 10-9 CWSIT messages

Specific message	Chapter
CWSIT0006E	► Chapter 11, “Messaging engine problem determination” on page 355 ► Chapter 15, “JMS application problem determination” on page 401
CWSIT0007W	► Chapter 11, “Messaging engine problem determination” on page 355 ► Chapter 15, “JMS application problem determination” on page 401
CWSIT0016E	► Chapter 15, “JMS application problem determination” on page 401
CWSIT0019E	► Chapter 15, “JMS application problem determination” on page 401
CWSIT0022E	► Chapter 15, “JMS application problem determination” on page 401
CWSIT0029I	► Chapter 16, “Messaging in a multiple messaging engine environment” on page 429
CWSIT0057E	► Chapter 24, “Foreign bus problem determination” on page 529
CWSIT0067E	► Chapter 24, “Foreign bus problem determination” on page 529
CWSIT0086E	► Chapter 15, “JMS application problem determination” on page 401 ► Chapter 24, “Foreign bus problem determination” on page 529
CWSIT0088E	► Chapter 15, “JMS application problem determination” on page 401
CWSIT0089E	► Chapter 15, “JMS application problem determination” on page 401
CWSIT0090E	► Chapter 15, “JMS application problem determination” on page 401

Specific message	Chapter
CWSIT0092E	► Chapter 15, “JMS application problem determination” on page 401
CWSIT0102E	► Chapter 15, “JMS application problem determination” on page 401
CWSIT0104E	► Chapter 15, “JMS application problem determination” on page 401

Table 10-10 CWSIV messages

Specific message	Relevant topics
CWSIV0775W	► Chapter 18, “Message-driven beans problem determination” on page 465

Table 10-11 CWSIQ messages

Specific message	Relevant topics
CWSIQ0017E	► Chapter 22, “WebSphere MQ link problem determination” on page 499

Table 10-12 CWSIZ messages

Specific message	Relevant topics
CWSIZ0002E CWSIZ0011E CWSIZ0020E CWSIZ0021E CWSIZ0045E CWSIZ0056E CWSIZ0057E CWSIZ0058E CWSIZ0059E	► Chapter 26, “Mediation problem determination” on page 565

Table 10-13 CWSJ messages

Specific message	Relevant topics
CWSJP9999E CWSJP0023E CWSJP0002E	► Chapter 20, “WebSphere MQ server problem determination” on page 481
CWSJR1181E CWSJR1192E	► Chapter 18, “Message-driven beans problem determination” on page 4656

Table 10-14 CWSOM messages

Specific message	Relevant topics
CWSOM1017E	► Chapter 14, “File store problem determination” on page 389
CWSOM1042E	► Chapter 14, “File store problem determination” on page 389

Table 10-15 CNTR messages

Specific message	Relevant topics
CNTR0020E	► Chapter 18, “Message-driven beans problem determination” on page 465

Table 10-16 CWPK messages

Specific message	Relevant topics
CWPKI0022E	► Chapter 25, “Default messaging provider security” on page 549

Table 10-17 DCSV messages

Message prefix	Reference
DCSV*xxxxx*	► *WebSphere Application Server V6.1: Workload Management Problem Determination*, REDP-4308

Table 10-18 HMGR messages

Message prefix	Reference
HMGR*xxxxx*	► *WebSphere Application Server V6.1: Workload Management Problem Determination*, REDP-4308

Table 10-19 J2CA messages

Specific message	Relevant topics
J2CA0164E	► Chapter 18, “Message-driven beans problem determination” on page 465
J2CA0052E	► Chapter 18, “Message-driven beans problem determination” on page 465
J2CA0137E	► Chapter 18, “Message-driven beans problem determination” on page 465

Table 10-20 WMSG messages

Specific message	Relevant topics
WMSG0902E	► Chapter 21, “WebSphere MQ configuration” on page 493
WMSG1603E	► Chapter 20, “WebSphere MQ server problem determination” on page 481 ► Chapter 21, “WebSphere MQ configuration” on page 493
WMSG1604E	► Chapter 21, “WebSphere MQ configuration” on page 493.
WMSG1605E	► Chapter 20, “WebSphere MQ server problem determination” on page 481 ► Chapter 21, “WebSphere MQ configuration” on page 493

Table 10-21 Other product messages

Specific message	Chapter
AMQ9202 AMQ9519 AMQ9520 AMQ9534 AMQ9558	► Chapter 22, “WebSphere MQ link problem determination” on page 499
DSRA8000E	► Chapter 13, “Message store with data store persistence” on page 369
TCPC0003E	► Chapter 11, “Messaging engine problem determination” on page 355

10.6 Guide to chapters by exception

Table 10-22 contains key exceptions that are discussed in the following chapters. When a message occurs followed by an exception, the message is considered the key indicator. Search for messages in 10.5, “Guide to chapters by message” on page 342 first. Use Table 10-22 if you do not find your key message.

Table 10-22 Guide to chapters by exception

Exception	Chapter
`ClassNotFoundException +` *`database error`*	► Chapter 17, “Clustering problem determination” on page 439
`com.ibm.db2.jcc.b.SqlException`	► Chapter 13, “Message store with data store persistence” on page 369
`com.ibm.db2.jcc.t2zos.y+Storage Allocation Error`	► Chapter 13, “Message store with data store persistence” on page 369

`com.ibm.ejs.jms.listener.MDBInvalidConfigException`	► Chapter 18, “Message-driven beans problem determination” on page 465
`com.ibm.mq.MQException`	► Chapter 20, “WebSphere MQ server problem determination” on page 481
`com.ibm.mqservices.MQInternalException`	► Chapter 19, “WebSphere MQ and MDBs” on page 477 ► Chapter 23, “JMS application with WebSphere MQ problem determination” on page 523
`com.ibm.ws.sib.msgstore.PersistenceException`	► Chapter 14, “File store problem determination” on page 389
`com.ibm.ws.sib.comms.server.ObjectStoreFullException`	► Chapter 15, “JMS application problem determination” on page 401
`com.ibm.ws.sib.msgstore.MessageStoreRuntimeException`	► Chapter 17, “Clustering problem determination” on page 439
`com.ibm.wsspi.sib.core.exception.SIRollbackException`	► Chapter 14, “File store problem determination” on page 389
`java.io.FileNotFoundException`	► Chapter 14, “File store problem determination” on page 389
`java.lang.ClassNotFoundException` + *database error*	► Chapter 13, “Message store with data store persistence” on page 369
`java.lang.Exception: De-reference of JMS provider's Reference failed - check provider is on classpath`	► Chapter 21, “WebSphere MQ configuration” on page 493
`java.sql.SQLException`	► Chapter 17, “Clustering problem determination” on page 439
`javax.jms.JMSException + CWSIA0241E`	► Chapter 15, “JMS application problem determination” on page 401
`javax.jms.InvalidDestinationException`	► Chapter 24, “Foreign bus problem determination” on page 529
`javax.jms.JMSException` + *MQ reason code*	► Chapter 23, “JMS application with WebSphere MQ problem determination” on page 523
`javax.jms.JMSExceptions + WebSphere MQ return code 2046`	► Chapter 21, “WebSphere MQ configuration” on page 493

`java.lang.UnsatisfiedLinkError: mqjbnd05 (Not found in java.library.path)`	► See PK40371: MQ_INSTALL_VERSION Environment Variable needed for support for BINDINGS mode connection to WebSphere MQ 5.3 at: `http://www-1.ibm.com/support/docview.wss?uid=swg1PK40371`
`javax.naming.NameNotFoundException`	► Chapter 18, "Message-driven beans problem determination" on page 465 ► Chapter 19, "WebSphere MQ and MDBs" on page 477
`javax.jms.InvalidDestinationException`	► Chapter 19, "WebSphere MQ and MDBs" on page 477
`com.ibm.ejs.jms.listener.MDBInvalidConfigException`	► Chapter 19, "WebSphere MQ and MDBs" on page 477
`JMSSecurityException`	► Chapter 25, "Default messaging provider security" on page 549
`SILimitExceededException`	► Chapter 16, "Messaging in a multiple messaging engine environment" on page 429
`SSLHandshakeException`	► Chapter 25, "Default messaging provider security" on page 549
Transaction rollback exceptions	► Chapter 13, "Message store with data store persistence" on page 369 ► Chapter 14, "File store problem determination" on page 389

10.7 Guide to chapters by symptom

Table 10-23 on page 351 contains key symptoms that are discussed in the following chapters. If you have not identified an error message or exception, use this table to identify your symptom.

Table 10-23 Guide to chapters by symptom

Chapter	Symptoms
Chapter 11, “Messaging engine problem determination” on page 355	► A messaging engine fails to start ► A messaging engine is active but fails to accept work ► In z/OS, a Control Region Adjunct (CRA) abends
Chapter 13, “Message store with data store persistence” on page 369	► Messaging engine fails to start ► Messaging engine is active but fails to accept work ► Error messages with a prefix of CWSIS. The message text indicates a problem with a data store, data source, or database. ► ClassNotFoundException that includes a database error message ► Rollback exceptions in the messaging application ► com.ibm.db2.jcc.b.SqlException
Chapter 14, “File store problem determination” on page 389	► Messaging engine fails to start. ► Applications experience transaction rollbacks ► java.io.FileNotFoundException exception ► com.ibm.ws.sib.msgstore.PersistenceException ► com.ibm.wsspi.sib.core.exception.SIRollbackException
Chapter 15, “JMS application problem determination” on page 401	► A JMS application is unable to create a connection to a service integration bus. ► A JMS application is unable to produce messages to the WebSphere Application Server default messaging provider ► JMS application unable to consume messages from the WebSphere Application Server default messaging provider. ► javax.jms.JMSException exceptions ► com.ibm.ws.sib.comms.server.ObjectStoreFullException exceptions ► Lost messages
Chapter 16, “Messaging in a multiple messaging engine environment” on page 429	► Messages are not found when expected on the queue ► Messages are not consumed from the queue when expected ► Application fails with an SILimitExceededException failure while sending messages to a queue. ► Application fails with an SILimitExceededException failure while sending messages to a topic space.
Chapter 17, “Clustering problem determination” on page 439	► One or more messaging engines do not start. ► Non-delivery or orphaning of messages. ► Multiple messaging engines are started on one cluster member while no messaging engines are started on others. ► The messaging engine is not starting on the preferred server. ► Response messages appear on a different partition than the request message (messages appear in an unexpected place). ► Orphan messages after a failover. ► Messages are not consumed from a partitioned destination. ► Messages do not arrive on a specified partition of a partitioned destination.

Chapter 18, "Message-driven beans problem determination" on page 465	► A message-driven bean application fails to start. ► A message-driven bean fails to connect to a messaging engine. ► A message-driven bean fails to consume messages. ► A message-driven bean fails during processing. ► Lost messages
Chapter 19, "WebSphere MQ and MDBs" on page 477	► The MDB application does not start. ► The MDB fails to connect to the MQ queue. ► The MDB application starts, but is not driven by messages on the WebSphere MQ destination. ► com.ibm.mqservices.MQInternalException exceptions ► javax.naming.NameNotFoundException for the destination ► javax.jms.InvalidDestinationException for the destination ► com.ibm.ejs.jms.listener.MDBInvalidConfigException for the destination
Chapter 20, "WebSphere MQ server problem determination" on page 481	► Messages are not processed via the WebSphere MQ server. Messages might appear to be lost. ► Unable to produce messages to a WebSphere MQ server destination. ► Unable to consume messages from a WebSphere MQ destination ► JMS messages arrive but appear corrupt ► The application is unable to connect to the WebSphere MQ server
Chapter 21, "WebSphere MQ configuration" on page 493	► JMS objects such as queue connection factories are defined, but do not appear in JNDI. ► java.lang.Exception: De-reference of JMS provider's Reference failed - check provider is on classpath ► Client application gets JMSExceptions when attempting to run in bindings mode with a stack trace containing the WebSphere MQ return code 2046.
Chapter 22, "WebSphere MQ link problem determination" on page 499	► The sender channel from WebSphere MQ Link fails to start. ► The receiver channel from WebSphere MQ fails to start ► Message flow problem from the service integration bus to WebSphere MQ. ► Message flow problem from WebSphere MQ to the service integration bus ► Messages flow across the WebSphere link but appear corrupt

Chapter 23, “JMS application with WebSphere MQ problem determination” on page 523	► A JMS application receives a JMSException when attempting to connect to WebSphere MQ. ► A JMS application receives a JMSException while attempting to create a producer for a given WebSphere MQ destination. ► A JMS application receives a JMSException while attempting to send a message. ► A JMS application appears to have sent the message successfully, but the message never arrives on the expected WebSphere MQ destination. ► A JMS Application receives a JMSException when attempting to create a consumer on for given WebSphere MQ destination.
Chapter 24, “Foreign bus problem determination” on page 529	► A messaging engine fails to start after configuring a foreign bus link ► A foreign bus link fails to start. ► JMS application gets an exception attempting to produce messages on a foreign destination.
Chapter 26, “Mediation problem determination” on page 565	► Messages are not being consumed the application. ► Messages are being consumed, but are unmediated. ► Messages are being mediated incorrectly. ► Messages are mediated, but slowly.
Chapter 25, “Default messaging provider security” on page 549	► Authentication problems connecting to the bus. ► Authorization problems connecting to the bus. ► Authorization problems accessing destinations. ► SSL problems connecting to the bus.

11

Messaging engine problem determination

Messaging engine problems usually fall into two categories: problems with starting the messaging engine and problems with connecting to the messaging engine. User applications designed to use the default messaging provider start independently from the messaging engine subsystem. However, they might become inactive or appear hung if the messaging engine is not running.

This chapter helps you distinguish between these two problem types, and it discusses in more detail problems with starting the messaging engine.

For information about problems that occur when the application attempts to connect to an active messaging engine, see Chapter 15, “JMS application problem determination” on page 401

11.1 Introduction

A messaging engine is a component of the WebSphere Application Server providing messaging functionality within a Service Integration Bus. For the application server to be messaging capable (using WebSphere default messaging) and to enable it to process messages locally, you must configure a messaging engine for it, and that messaging engine must be in a *Started* state.

For information about messaging engine configuration, see *Configuring Messaging Engines at:*

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/tasks/tjj0050_.html`

11.1.1 Symptoms of a messaging engine problem

Generally, messaging engine problem determination can be categorized into the following two areas:

- Messaging engine start-up problems
- Application to messaging engine connectivity

Symptoms addressed in this chapter include:

- A messaging engine fails to start
- A messaging engine is active but fails to accept work
- In z/OS, a Control Region Adjunct (CRA) abends

11.2 Verify the messaging engine is started

The first step in diagnosing any messaging problem is to determine the state of the messaging engines on the bus. Using 10.2.3, "Determine your messaging engine status" on page 328, determine which of the following states applies to your message engine.

The states and what they mean are:

- `Unavailable`

  The message engine cannot be started. Messaging engine initialization is dependent on the bus members' application servers running the SIB Service (Service Integration Bus Service). While this service might be enabled at any time, and it is automatically enabled for newly added bus members, it is initialized during server start-up.

In a newly-configured messaging engine, the new messaging engine might remain in the Unavailable state until you either start or restart its application servers.

- `Stopped`

  A Stopped state might not be an indication of a problem. For example, an administrator might have stopped the messaging engine to perform an administrative task.

  You can restart the messaging engine from the administrative console; or, you can wait for it to automatically start during the next application server restart.

- `Started`

  In the messaging engine is in Started state, the underpinning messaging configuration is operational. Your next step in diagnosing the problem is to determine whether the application was able to connect to the messaging engine.

11.3 Analyze diagnostics

Problems encountered during messaging engine start-up are written to the application server log files. Problems encountered when connecting to a messaging engine are captured and reported by the JMS application. To collect this documentation, see 10.3, “Collect diagnostics” on page 339.

To analyze the problem:

- If you experienced a Control Region Adjunct (CRA) abend in z/OS, see 11.4.2, “Incorrect Control Region Adjunct (CRA) ownership (z/OS)” on page 362.
- Analyze the SystemOut log for the application server running the messaging engine to determine if the messaging engine started correctly.
- If the messaging engine started correctly, examine the logs associated with the application to determine if it was able to connect to the messaging engine.

This chapter addresses some specific error messages and their root causes. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 27, “The next step” on page 581.

11.3.1 Analyze SystemOut log for the messaging engine

In the SystemOut log, search for error messages beginning with `CWS`, `TCPC`, `DCSV`, or `HMGR`. Start with the most recent log entries first and work, in order, back through older entries. Pay attention to all CWSI messages, which contain information about the messaging engine startup.

Look for any of these messages:

- CWSID0016I

  Indicates messaging engine state changes, including `Joined`, `Starting`, `Started`, `Stopping`, `Stopped`. An application can connect to a messaging engine only when the engine is in the `Started` state.

- CWSIS0002E

  Indicates an error occurred while starting the messaging engine. Messages encapsulated within this error and error messages that follow provide more information about the problem.

  - A CWSIS0002E message with an embedded CWSIS message indicates a *message store* problem. The embedded message text indicates a problem with a data store, data source, or database.

    See Chapter 12, "Message store problem determination" on page 365.

  - A CWSIS0002E message with an embedded CWSOM message indicates a *file store configuration* problem. The CWSOM error message indicates a problem with a file store or log file.

    See Chapter 14, "File store problem determination" on page 389.

- TCPC0003E

  The following message during messaging engine startup indicates a *port conflict*. If this occurs, the messaging engine will start but will not accept work. See Section 11.4.1, "Service integration port conflicts" on page 362.

  ```
  TCPC0003E: TCP Channel SIB_TCP_JFAP initialization failed.  The
  socket bind failed for host * and port 7276.  The port may already
  be in use.
  ```

- DCSV*xxxxx* and HMGR*xxxxx* messages

  Another common cause of a messaging failing to start is problems with the distributed environment, usually reflected in DCS (distribution and consistency services) messages in the SystemOut log. For help with these types of problems see *WebSphere Application Server V6.1: Workload Management Problem Determination*, REDP-4308.

Normal messaging engine start-up

To evaluate a messaging engine start-up problem, you need to be familiar with how a normal start-up appears in the SystemOut log. Example 11-1 is an extract from an application server log file; it shows the key entries associated with normal messaging engine start-up when the messaging engine is configured to use a file store for persistence.

Example 11-1 Messaging engine start up with file store

```
WsServerImpl  A   WSVR0001I: Server server1 open for e-business
:
SibMessage    I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Joined.
SibMessage    I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Starting.
:
SibMessage    I   [:] CWSIS1566I: A single previous owner was found in
the messaging engine's file store, ME_UUID=1D842649652FA294,
INC_UUID=0E040E040F8E07CD
SibMessage    I   [:] CWSIS1563I: The messaging engine,
ME_UUID=1D842649652FA294, INC_UUID=0D800D800FA453BF, has acquired an
exclusive lock on the file store.
:
SibMessage    I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Started.
```

Example 11-2 is an extract from an application server log file that shows the key entries associated with normal messaging engine start-up where the messaging engine is configured to use a data store for persistence. See Chapter 12, “Message store problem determination” on page 365 for more information about messaging engine persistence options.

Example 11-2 Messaging engine data store

```
SibMessage   I   [:] CWSID0021I: Configuration reload is enabled for
bus Bus1.
SibMessage   I   [:] CWSIS1568I: Messaging engine
WASNode01.server1-Bus1 is using a data store.
SibMessage   A   [:] CWSIC2001I: Messaging connections are being
accepted.
SibMessage   I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Joined.
SibMessage   I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Starting.
```

```
SibMessage   I   [:] CWSIS1538I: The messaging engine,
ME_UUID=628531B848F5CBFF, INC_UUID=07FA07FAFAFCFDA2, is attempting to
obtain an exclusive lock on the data store.
SibMessage   I   [:] CWSIS1543I: No previous owner was found in the
messaging engines data store.
SibMessage   I   [:] CWSIS1537I: The messaging engine,
ME_UUID=628531B848F5CBFF, INC_UUID=07FA07FAFAFCFDA2, has acquired an
exclusive lock on the data store.
SibMessage   I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Started.
```

Problems during messaging engine startup

When a messaging engine fails to start, the messages shown in Example 11-3 generally occur and are commonly grouped together. The resulting messaging engine state is `Unavailable`.

Example 11-3 Messaging engine startup problem

```
... CWSIS0002E: The messaging engine encountered an exception while
starting. Exception: ...
:
CWSID0027E: Messaging engine <MessagingEngineName> cannot be restarted
because a serious error has been reported.
CWSID0016I: Messaging engine <MessagingEngineName> is in state Stopped.
CWSID0016I: Messaging engine <MessagingEngineName> is in state Joined.
```

The CWSIS0002E message provides the primary source of problem determination information for messaging engine startup problems. This is a generic message; however, the encapsulated message text normally contains information that can help you identify the root cause of the problem.

If the messages indicate the messaging engine started correctly, continue with section 11.3.2, "Analyze the application logs" on page 360 to determine if the application was able to connect to the messaging engines.

11.3.2 Analyze the application logs

JMS applications that use the default messaging provider connect to a messaging engine, whether they run inside the application server or externally in a client container. This process is abstracted with the use of a connection factory, and the initial connection request, or *bootstrap*, might simply serve to redirect the application to another messaging engine on the bus.

Find errors that occur when the application connects to the messaging engine in the application logs:

- If the messaging engine experiences port conflicts, it starts but does not accept work. You might see these messages:

  ```
  CWSIT0007W: It is not possible to contact the bootstrap server at
  localhost:7276:BootstrapBasicMessaging because of exception:
  com.ibm.websphere.sib.exception.SIResourceException:

  CWSIC1001E: A client attempted to connect with a remote messaging
  engine (localhost:7276 - 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 127.0.0.1, port 7276
  cannot be established... javax.jms.JMSException:

  CWSIA0241E: An exception was received during the call to the method
  JmsManagedConnectionFactoryImpl.createConnection:
  com.ibm.websphere.sib.exception.SIResourceException:

  CWSIT0006E: It was not possible to contact any of the specified
  bootstrap servers. Please see the linked exception for further
  details. Bootstrap connections were attempted to:
  [localhost:7276:BootstrapBasicMessaging].
  ```

  For more on port conflicts see 11.4.1, “Service integration port conflicts” on page 362.

- When the connection process for a J2EE application fails, a generic JMSException is generated on a createConnection, createQueueConnection or createTopicConnection:

  ```
  javax.jms.JMSException: CWSIA0241E
  ```

  If you see this type of exception, see Chapter 15, “JMS application problem determination” on page 401.

11.4 Causes and solutions

Common conditions that can prevent a messaging engine from starting or that can keep it from accepting work include:

- Service integration port conflicts
- Incorrect Control Region Adjunct (CRA) ownership (z/OS)

11.4.1 Service integration port conflicts

When a messaging engine is started but does not accept work, look for indications of a port conflict during startup. You might see a message such as the following in the SystemOut log for the messaging engine server:

```
TCPC0003E: TCP Channel SIB_TCP_JFAP initialization failed.  The socket
bind failed for host * and port 7276.  The port may already be in use.
```

A JMS client might experience this error:

Example 11-4 JMS client symptom of a port conflict

```
[:] CWSIT0007W: It is not possible to contact the bootstrap server at
localhost:7276:BootstrapBasicMessaging because of exception:
com.ibm.websphere.sib.exception.SIResourceException: CWSIC1001E: A
client attempted to connect with a remote messaging engine
(localhost:7276 - 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 127.0.0.1, port 7276 cannot be
established...
javax.jms.JMSException: CWSIA0241E: An exception was received during
the call to the method
JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It was
not possible to contact any of the specified bootstrap servers. Please
see the linked exception for further details. Bootstrap connections
were attempted to: [localhost:7276:BootstrapBasicMessaging].
```

This is a configuration error. If you see this message, review the ports assigned to the application server to ensure that there are no conflicts with the following ports:

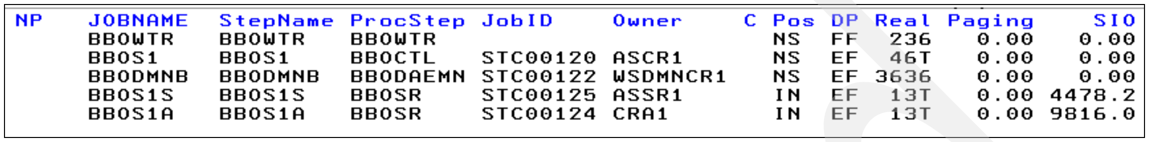
- Service Integration Bus (SIB) port
- SIB secure port
- SIB MQ interoperability port
- SIB MQ interoperability secure port

If you suspect that there might be a port conflict, you can use a command, such as **netstat**, to view the ports that are currently in use.

11.4.2 Incorrect Control Region Adjunct (CRA) ownership (z/OS)

An installation configuration error can result in not assigning the Adjunct region its own CRA1 ID for the Started Task. This result is not apparent until the CRA region is required. The symptom of this problem is a CRA abends due to incorrect ownership.

Figure 11-1 illustrates correct ownership for the task BBOS1A (last line). For an incorrectly created CRA region you would see ownership by the Servant Region (SR); that is, ASSR1.



```
NP     JOBNAME   StepName  ProcStep  JobID     Owner       C  Pos  DP  Real  Paging     SIO
       BBOWTR    BBOWTR    BBOWTR                            NS   FF   236    0.00    0.00
       BBOS1     BBOS1     BBOCTL    STC00120  ASCR1         NS   EF   46T    0.00    0.00
       BBODMNB   BBODMNB   BBODAEMN  STC00122  WSDMNCR1      NS   EF  3636    0.00    0.00
       BBOS1S    BBOS1S    BBOSR     STC00125  ASSR1         IN   EF   13T    0.00  4478.2
       BBOS1A    BBOS1A    BBOSR     STC00124  CRA1          IN   EF   13T    0.00  9816.0
```

Figure 11-1 BBOS1A ownership

If you determine that the CRA is abending due to incorrect ownership:

1. Enter the ISPF customization dialog in option 6. For example:

   `EX 'HLQ.SBBOCLIB(BBOWSTRT)' 'APPL(BBO) LANG(ENUS)' )`

2. Choose the option for **Create stand-alone Application Server nodes**.
3. Load your application server saved configuration customization variables.
4. Go into **Define Variables** and select the **Server Customization** option.
5. Page through the variables until you find the place where you can select the user ID and procedure for the CRA.
6. Change the user ID to `CRA1`.
7. Choose the option to generate the customization jobs, and then save the variables.
8. Rerun job BBOCBRAJ, which generates the RACF® commands used to create the user ID and started task definitions for the CRA. These are found in member BBOWBRAK.
9. Extract the commands relevant to the CRA from member BBOWBRAK.
10. If you had not created the ID before, you would run all the commands relevant to CRA, changing the RDEFINE to RALTER on the started task definition. For example, see Example 11-5.

Example 11-5 Create the user ID

```
ADDUSER CRA1 DFLTGRP(WSSR1) OMVS(UID(2433)
HOME(/var/WebSphere/home/WSSR1) PROGRAM(/bin/sh)) NAME('WAS APPSVR
ADJUNCT') NOPASSWORD NOOIDCARD

ALU CRA1 OMVS(FILEPROC(10000))
```

```
CONNECT CRA1 GROUP(WSCFG1)
PERMIT CB.*.WCLxxxx.* CLASS(SERVER) ID(CRA1) ACC(READ)

PERMIT CB.*.WCLxxxxADJUNCT.* CLASS(SERVER) ID(CRA1) ACC(READ)
RALTERSTARTEDprocname.*STDATA(USER(CRA1)GROUP(WSCFG1)TRACE(YES))
SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

If you already had defined the ID, you just need to run the command in Example 11-6.

Example 11-6 Alter the user ID

```
RALTERSTARTEDprocname.*STDATA(USER(CRA1)GROUP(WSCFG1)TRACE(YES))
SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

11.5 Validate the solution

Restart the messaging engine and bus member (if applicable). Check the messaging engine state to make sure it starts.

Message store problem determination

The message store holds the state of your WebSphere Application Server default messaging system, both persistently on a storage device and in memory at runtime. During configuration of the WebSphere Application Server, you can choose between these two configuration options for storing persistent message data:

- File store. Introduced in WebSphere V6.1, this mechanism uses flat files on a local or remote file system to store all persistent data.
- Data store. This method lets you use an existing relational database management system (RDBMS), such as IBM DB2, to store all persistent data.

Problems might occur when a messaging engine attempts to access its persistent message store. The first step in any message store problem investigation is to identify the type of message store persistence in use.

12.1 Determine message store type

The simplest method to check the message store type in use is by using the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.
4. Click on the messaging engine name.
5. On the right side of this page, click **Message store** to open the configuration page for the message store and its persistence mechanism.

   The type of message store is listed at the top of this page. For example:

   `Service integration → Buses → Bus_name → Messaging engines → Messaging_Engine_Name → Message_Store_Type`

   Where *Message_Store_Type* is listed as either **File store** or **Data store**.

Alternatively, you can determine the message store persistence type by examining the SystemOut log file. For a given messaging engine, one of the following entries is present in the application server log file:

- `SibMessage I [:] CWSIS1569I: Messaging engine Messaging_Engine_Name is using a file store.`
- `SibMessage I [:] CWSIS1568I: Messaging engine Messaging_Engine_Name is using a data store.`

Message store problems are written to the SystemOut log files. If an unexpected exception is caught during the startup of your message store, it will generally be enclosed as part of a CWSIS0002E error message. For example:

```
SibMessage E [:] CWSIS0002E: The messaging engine encountered an
exception while starting. Exception:
```

The contents of this message can help you (or IBM service personnel) determine the root cause of the problem. Messages and exceptions such as this one are used as reference points throughout this material to enable quicker determination of the problem you are experiencing. Always check your logs for messages and exception information before starting the problem determination exercise so you have the best possible chance of success.

12.2 Where to go from here

If your messaging engine has encountered a problem with its message store and it is configured to use *file store persistence*, see Chapter 14, “File store problem determination” on page 389.

If your messaging engine has encountered a problem with its message store and is configured to use *data store persistence*, see Chapter 13, “Message store with data store persistence” on page 369.

13

Message store with data store persistence

This chapter addresses problems with messaging engines that use a data store for message persistence.

If you are using a file store, see Chapter 14, “File store problem determination” on page 389. If you are not sure which mechanism you are using for persisting messages, see 12.1, “Determine message store type” on page 366.

13.1 Introduction to data stores

A data store uses an existing RDBMS to store all persistent data used by the WebSphere Application Server default messaging system. It lets you incorporate your messaging data into an existing data storage infrastructure.

These resources can help you understand data stores:

- For a description of data stores, see *Data stores* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.doc/concepts/cjm0410_.html

- For a detailed description of the data store locking, see *Data store exclusive access locking* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc\concepts\cjm0450_.html

- For information about setting up a data store in DB2 for z/OS see Chapter 9 of *Architecting High Availability Using WebSphere V6 on z/OS*, SG24-6850.

  http://www.redbooks.ibm.com/abstracts/sg246850.html

- For explanation of a specific DB2 z/OS error messages searching for the message in the IBM Information Management Software for z/OS Solutions Information Center at:

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp

13.1.1 Symptoms of a data store problem

Symptoms of a problem with the messaging engine data store include:

- Messaging engine fails to start
- Messaging engine is active but fails to accept work
- Error messages with a prefix of CWSIS. The message text indicates a problem with a data store, data source, or database.
- ClassNotFoundException that includes a database error message.
- Rollback exceptions in the messaging application
- com.ibm.db2.jcc.b.SqlException

13.2 Verify system integrity

If you a using a database as the messaging data store and you are having problems that appear related to the data store, first verify the following information, which are described in more detail below:

- Database status
- Database configuration
- Database connectivity
- Messaging engine state

13.2.1 Verify database status

A messaging engine configured to use a data store relies on connectivity to an underlying RDBMS. Your RDBMS is separate from your WebSphere system (unless you use the default Derby JDBC™ provider in a single server), and it might be in a disjointed state. In any case, if you suspect connectivity to the database could be the problem you should check the RDBMS to make sure that it:

- It is running satisfactorily
- It has network connectivity to your WebSphere server machine
- It is set up to accept connections from your WebSphere server

13.2.2 Verify database configuration

Consider these requirements regarding your data stores:

- The underlying database that is used by a data store must have a page size of at least 4K to hold the tables created by the system.
- DB2 for z/OS does not support the **create tables automatically** option. On DB2 for z/OS ,the DDL for your messaging engine tables must be created using the sibDDLGenerator tool, and then run by your database administrator to manually create the required tables before starting the messaging engine for the first time.

  For further information about setting up a data store in DB2 for z/OS, see Chapter 9 of *Architecting High Availability Using WebSphere V6 on z/OS*, SG24-6850.

13.2.3 Verify connectivity to the database

You can test the connection to the database using the administrative console. This test is helpful in discovering network connectivity problems to the database and configuration problems if this is a newly defined database.

For information about testing connections using the administrative console, see *Testing a connection with the administrative console* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tdat_testcon.html

The **Test connection** button in the administrative console is not a 100% reliable method of testing connectivity to your database. The authentication values of the data source might not match those of the messaging engine and environment variables might need adjusting. For more information, see *Test connection service* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cdat_testcon.html

13.2.4 Verify the messaging engine is started

Use the process in 10.2.3, “Determine your messaging engine status” on page 328 to determine the state of your messaging engine:

- If the messaging engine is not in Started state, attempt to start it. If the messaging engine will not start, see 13.3, “Messaging engine fails to start” on page 372
- If the messaging engine is in Started state, see 13.4, “Messaging engine not accepting work” on page 385

13.3 Messaging engine fails to start

A messaging engine that fails to start is often the first visible symptom of a data store configuration problem. Look in the SystemOut log for indications of why the messaging engine fails.

13.3.1 Analyze the SystemOut log for the messaging engine

Look for any CWSIS*xxxxx* messages. In particular, pay attention to error messages, but information and warning messages are useful as well. Look for any ClassNotFoundException errors that include the data source or JDBC driver for the messaging data store.

The following list contains specific error messages that apply to data store problems. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 27, “The next step” on page 581.

Database connectivity errors

The following errors indicate problems with database connectivity:

- If you see either of these messages, see 13.3.2, “Data source not found” on page 375:
  - `SibMessage I [:]` **`CWSIS1514E:`** `A data source JNDI name has not been specified.`
  - `SibMessage I [:]` **`CWSIS1524E:`** `Data source jdbc/com.ibm.ws.sib/testNode01.server1-default not found.`
- If you see this message, either in the SystemOut log or when you try a Test Connection function, see 13.3.3, “Data source ClassNotFoundException” on page 377:
  - **java.lang.ClassNotFoundException** for a data source, for example:

    `java.lang.ClassNotFoundException: DSRA8000E: No jar or zip files found in` *`JDBC_DRIVER_PATH`*
- If you see this message, see 13.3.4, “Data source connection pool constraint” on page 377.
  - `SibMessage I [:]` **`CWSIS1522E:`** `The messaging engine's request for a database connection timed out.`

Database locking errors

To ensure data integrity in a data store the messaging engine takes an exclusive lock on one of its tables at startup. If this lock cannot be obtained the messaging engine will not start.

For more detailed information about data store exclusive access locking please read the *Data store exclusive access locking* in the Information Center at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjm0450_.html`

These error messages indicate problems with database locking:

- CWSIS1519E indicators a locking problem; however, the root cause might vary depending on whether you see this message after a failover has occurred or at initial startup of the messaging engine:

  – `SibMessage I [:] CWSIS1519E: Messaging engine testNode01.server1-default cannot obtain the lock on its data store, which ensures it has exclusive access to the data.`

  If you see this message and *no failover has occurred*, see 13.3.5, “Messaging engine fails to obtain a lock” on page 378.

  If you see this message *after a failover situation*, see 13.3.6, “Messaging engine fails to obtain lock on failover” on page 378.

- If you see CWSIS1535E, see 13.3.7, “Messaging engine unique ID does not match data store” on page 379:

  – `SibMessage E [:] CWSIS1535E: The messaging engine's unique id does not match that found in the data store.`

    This message is usually accompanied by these messages:

    `SibMessage I [:] CWSIS1538I: The messaging engine, ME_UUID=ME2, INC_UUID=INC1, is attempting to obtain an exclusive lock on the data store.`

    `SibMessage I [:] CWSIS1545I: A single previous owner was found in the messaging engine's data store, ME_UUID=ME1, INC_UUID=INC1`

Errors specific to DB2 for z/OS

Some common errors are specific to using DB2 for z/OS to host a data store. You might need to work closely with your DBA and SYSPROG when working through these problems.

If you are using DB2 for z/OS to host the data store, look for these messages in the SystemOut log:

Example 13-1 DB2 for z/OS data source exception

```
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:
exception_information
```

Information specific to the problem is contained in the *exception_information* of the CWSIS1501E message. The following are possible exceptions:

- If you see this exception, see 13.3.8, “Data source exception: SQLCODE -471” on page 381:

```
com.ibm.db2.jcc.b.SqlException: DB2 SQL error: SQLCODE: -471,
SQLSTATE: 55023, SQLERRMC: SYSIBM.SQLTABLES;00E7900C
```

- If you see this exception, see 13.3.9, “Data source exception: unavailable resource” on page 382:

```
com.ibm.db2.jcc.b.SqlException: UNSUCCESSFUL EXECUTION CAUSED BY AN
UNAVAILABLE RESOURCE. REASON 00E7009A, TYPE OF RESOURCE 100, AND
RESOURCE NAME TEMP DATABASE
```

- If you see this exception, see 13.3.10, “Data source exception: failure to load native library” on page 383:

```
com.ibm.db2.jcc.b.SqlException: Failure in loading T2 native library
db2jcct2DSRA0010E: SQL State = null, Error Code = -99, 999
```

- If you see this exception, see 13.3.11, “Data source exception: storage allocation error” on page 384:

```
com.ibm.db2.jcc.t2zos.y:
[IBM/DB2][T2zos/2.5.48]T2zosPreparedStatement.readPrepareDescribeOut
put_:processDescribeOutput:1563:Storage Allocation Error at
com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink.readDataFromPer
sistence(AbstractItemLink.java:2487) at
com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink._restoreItem(Ab
stractItemLink.java:639)
```

13.3.2 Data source not found

Access to your chosen RDBMS is provided to the message store through a JDBC data source defined in your application server configuration. If this data source cannot be found by the message store at runtime, there are two error messages that might be output to notify you of the problem.

The symptoms of this problem are that the messaging engine will not start, and one of these messages appears in SystemOut:

- `SibMessage I [:] CWSIS1514E: A data source JNDI name has not been specified.`
- `SibMessage I [:] CWSIS1524E: Data source data_source_name not found.`

CWSIS1514E

This message is refers to the data source field on the administrative console panel for your message store.

Solution

To check the value in the data source field for the messaging engine to make sure that it is not missing:

1. Select **Service integration** → **Buses** and click the bus name to open the details page.
2. Click **Messaging engines** and click on the name of the messaging engine to open the details page.
3. Click **Message Store** under Additional Properties.
4. Validate that the JDBC name listed under the **Data source JNDI name** heading matches the data source you have configured.

CWSIS1524E

This message indicates that the message store cannot find the data source name that it has been provided within the JNDI name space.

The most common cause for this problem is that the data source for the messaging data store is not defined at a scope visible to the messaging engine.

For example, if the messaging engine is in cell3/node2/server1, the data source might be defined in either server1 or node2 or cell3 to be visible for use by the data store.

Within a highly available environment, there are additional factors to consider when you create data sources, as you might have a messaging engine that can run on more than one server.

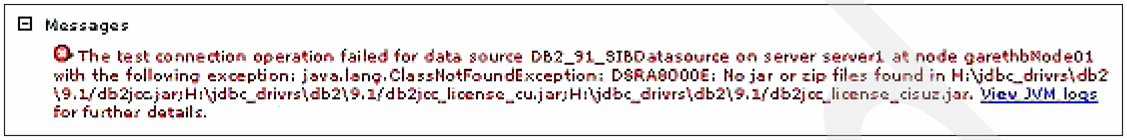
For example, if you have a single messaging engine which can fail over between two servers: cell3/node2/server1 and cell3/node2/server2, you need to define the data source at a scope visible to both servers. In this case, a suitable scope could be either node2 or cell3. A common mistake is to create the data source with a scope of server1. Initially, the configuration will appear to work, when the messaging engine starts on server1. However, when the messaging engine fails-over to server2, server2 will not be able to locate the appropriate data source.

Solution

Ensure that the definition of the data source is in a scope that is visible to the messaging engine (or engines) that are configured to use it.

13.3.3 Data source ClassNotFoundException

The symptoms of this problem are that the messaging engine will not start and you receive a data source ClassNotFound Exception message. This exception in Figure 13-1 can also appear when you try a Test Connection to the database from the administrative console.



```
The test connection operation failed for data source DB2_91_SIBDatasource on server server1 at node garethbNode01
with the following exception: java.lang.ClassNotFoundException: DSRA8000E: No jar or zip files found in H:\jdbc_drivrs\db2
\9.1/db2jcc.jar;H:\jdbc_drivrs\db2\9.1/db2jcc_license_cu.jar;H:\jdbc_drivrs\db2\9.1/db2jcc_license_cisuz.jar. View JVM logs
for further details.
```

Figure 13-1 ClassNotFoundException on Test Connection

This problem is commonly caused by the content or scoping of your WebSphere Application Server environment variables.

Solution

Validate the paths referenced in the exception information to help narrow the problem. For example the following path is part of the default value for a new DB2 JDBC provider:

${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar

If the value of ${DB2UNIVERSAL_JDBC_DRIVER_PATH} is not set then the application server will not be able to load the classes required to run any data source configured to use this provider. In cases where the value is set correctly, you must check to see that it is defined at a scope visible to the server in which the provider is defined.

The simplest way to check a WebSphere Application Server environment variable is to use the administrative console. Select **Environment → WebSphere Variables.**

13.3.4 Data source connection pool constraint

The symptoms of this problem are that the messaging engine will not start and the following message appears in SystemOut:

```
SibMessage I [:] CWSIS1522E: The messaging engine's request for a
database connection timed out.
```

This message indicates that the messaging engine has encountered a limitation in the current configured size of the connection pool used by the data source.

Solution

The recommended minimum size of connection pool for your messaging engine's data source is at least 50. This setting allows a large number of concurrent threads to carry our persistent messaging work.

When increasing the size of your connection pool make sure to check your RDBMS documentation to ensure your database supports the required number of concurrent sessions.

For more information, see *Tuning the JDBC data source of a messaging engine* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.iseries.doc\tasks\tjm0230_.html

13.3.5 Messaging engine fails to obtain a lock

In this situation, the messaging engine fails to start with CWSIS1519E.

In a high availability configuration (the default for clustered messaging engines) where you have multiple instances of a messaging engine configured to fail over, a race could occur between two instances trying to get the lock on their shared data. In this case, the failing instance produces the following message when it fails to acquire its lock:

```
SibMessage I [:] CWSIS1519E: Messaging engine
testNode01.server1-default cannot obtain the lock on its data store,
which ensures it has exclusive access to the data.
```

Solution

Check that no other instances of the messaging engine are running in the cluster.

13.3.6 Messaging engine fails to obtain lock on failover

When a messaging engine fails over, the new messaging engine instance (ME2) attempts to gain its exclusive lock on the database tables as normal. However, the previous instance of the messaging engine (ME1) might not have failed in a controlled manner, which can lead to the lock that it held not being released correctly. Therefore, when ME2 tries to connect to the tables, the pre-existing lock from ME1 might stop it from getting a lock. In this case, you see this message in your logs:

```
SibMessage I [:] CWSIS1519E: Messaging engine
testNode01.server1-default cannot obtain the lock on its data store,
which ensures it has exclusive access to the data.
```

The usual reason for ME1's lock to last longer than the messaging engine instance is the database waiting for the connection between ME1 and the database to timeout.

For example, suppose that the dropped connection timeout is set to five minutes in the database server but it only takes two minutes for ME2 to start after a failover. This situation would leave a time period of three minutes in which ME2 cannot yet gain a lock on the database tables, even though ME1 no longer exists.

This situation is only a minor problem because ME2 continues to try and gain the lock until the database releases the lock that ME1 held, and things can continue as normal.

If, however, the default timeout values in your database system are not small, you can see how easily this outage could become a major one while ME2 is waiting for the lock to be released.

Another version of this problem has been seen when running a database server in a UNIX environment (including Linux). In this case, it is not the database timeout which is causing the problem; it is the operating system level TCP_KEEPALIVE timeout value. The default for this value can be in multiples of hours which would have a major affect on your systems ability to fail over in a timely fashion.

Solution

Check that the connection timeout values in your database server configuration match your expectations regarding failover time. Also check the values for the operating systems TCP timeouts to make sure they allow failover in a timely fashion.

13.3.7 Messaging engine unique ID does not match data store

The symptoms of this problem are that the messaging engine will not start and the following message appears in SystemOut:

```
SibMessage E [:] CWSIS1535E: The messaging engine's unique id does not
match that found in the data store.
```

This problem generally occurs when a messaging engine detects that it does not have exclusive ownership of a data store.

There are two potential causes: ME_UUID mis-match or INC_UUID mis-match.

ME_UUID mis-match

Example 13-2 illustrates the typical error message reported in the SystemOut log files:

Example 13-2 ME_UUID mis-match

```
SibMessage I [:] CWSIS1538I: The messaging engine, ME_UUID=ME2,
INC_UUID=INC1, is attempting to obtain an exclusive lock on the data
store.
SibMessage I [:] CWSIS1545I: A single previous owner was found in the
messaging engine's data store, ME_UUID=ME1, INC_UUID=INC1
SibMessage E [:] CWSIS1535E: The messaging engine's unique id does not
match that found in the data store. ME_UUID=ME2, ME_UUID(DB)=ME1
```

Referring to the messages in Example 13-2, you see that the ME_UUIDs quoted in the CWSIS1535 message do not match. Messaging engine ME2 is trying to take an exclusive lock on tables that are owned by ME1. This will never succeed.

The most likely causes for this problem are failing to clean up the data from a deleted messaging engine or miss-configuration. One of these conditions is likely occurring:

- A new messaging engine (ME2) is attempting to use the same database and schema information as a previously deleted messaging engine (ME1), without the data for ME1 being deleted.
- ME2 is attempting to use the same schema name as ME1 in the same database.
- ME2 is mistakenly using the data source defined for ME1.

Solution

Make sure that the data source you have configured for your data store references the correct database instance and that the schemas being used by each messaging engine are unique.

INC_UUID mis-match

Example 13-3 illustrates the typical error message reported in the application server log files.

Example 13-3 INC-UUID mis-match

```
SibMessage I [:] CWSIS1538I: The messaging engine, ME_UUID=ME1,
INC_UUID=INC2, is attempting to obtain an exclusive lock on the data
store.
SibMessage I [:] CWSIS1545I: A single previous owner was found in the
messaging engine's data store, ME_UUID=ME1, INC_UUID=INC1
```

```
SibMessage E [:] CWSIS1535E: The messaging engine's unique id does not
match that found in the data store. ME_UUID=ME1, INC_UUID=INC1,
ME_UUID(DB)=ME1, INC_UUID(DB)=INC2
```

Referring to the messages in Example 13-3 on page 380, you see that the INC_UUIDs quoted in the CWSIS1535E message do not match. An INC_UUID is used to identify a particular instance of a messaging engine. For example if a messaging engine is configured to fail over between two servers, as in the example given here, there are one ME_UUID (ME1) and two INC_UUIDs (INC1 and INC2). To make sure that only one of these messaging engines can access the data store at any one time, the INC_UUID is checked to determine which instance is currently in control.

The likely cause of this problem is a drop in communication between the database and the messaging engine. This would lead to:

1. ME1,INC1 starts and gets a lock on its data store.
2. ME1,INC1 loses its connection to the database and hence its lock.
3. ME1,INC1 retries to get its lock, but the database is still down.
4. ME1,INC2 is started by the HA manager to replace ME1,INC1.
5. ME1,INC2 find the database is available and acquires the lock.
6. ME1,INC1 also finds the database is now available but notices that another instance has acquired the lock and outputs CWSIS1535.

The root cause of this problem is the loss of connectivity to the database.

Solution

Check your network connectivity and database server logs to determine why the problem occurred.

13.3.8 Data source exception: SQLCODE -471

This information is for DB2 for z/OS only.

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 13-4.

Example 13-4 Data source exception: SQLCODE -471

```
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.db2.jcc.b.SqlException: DB2 SQL error: SQLCODE: -471, SQLSTATE:
55023, SQLERRMC: SYSIBM.SQLTABLES;00E7900C
```

This problem can occur if you have configured your messaging engine to use the DB2 Universal JDBC Driver to connect to DB2 on z/OS. The JDBC driver needs to execute stored procedures that are not available.

Solution

Work with your database administrator to ensure the following requirements have been met:

- DSNUTILS is configured to use WLM.
- DSNUTILS is running APF authorized.
- The WLM environment you are trying to use is defined to WLM and is in an available state.
- DB2 has RACF authority to use WLM.

Refer to the following resources for further guidance on this problem:

- *Code 00E7900C*

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.mc/msgs/code00e7900c.html

- *DB2-supplied stored procedures -Invoking utilities as a stored procedure (DSNUTILS)*

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.ugref/db2stpr.htm

13.3.9 Data source exception: unavailable resource

DB2 for z/OS only

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 13-5.

Example 13-5 Data source exception: unavailable resource

```
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.db2.jcc.b.SqlException: UNSUCCESSFUL EXECUTION CAUSED BY AN
UNAVAILABLE RESOURCE. REASON 00E7009A, TYPE OF RESOURCE 100, AND
RESOURCE NAME TEMP DATABASE
```

This problem indicates that either the TEMP database is not defined or that its page size is not suitable for your use.

Solution

Define a suitable TEMP database for the DB2 Universal JDBC driver so that it will work correctly with DB2 for z/OS.

Work with the database administrator to ensure the following has been done:

- A TEMP database is created for each member of a data sharing group.
- The TEMP database has a 32k page size by defining it in a 32K buffer pool.

Refer to the following resources for further guidance on this problem:

- *Code 00E7009A*

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.mc/msgs/code00e7009a.html

13.3.10 Data source exception: failure to load native library

DB2 for z/OS only

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 13-6:

Example 13-6 Data source exception: unavailable resource

```
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.db2.jcc.b.SqlException: Failure in loading T2 native library
db2jcct2DSRA0010E: SQL State = null, Error Code = -99, 999
```

The problem is that the DB2 native libraries cannot be found by the DB2 JDBC driver.

Solution

The DB2 libraries for your WebSphere Application Server installation might have been placed in the linklist or //STEPLIB DD concatenation for the WebSphere Application Server for z/OS address spaces that will use JDBC. In some installations a combination of techniques are used.

If the DB2 libraries have been placed in the linklist or steplib, refer to this resource for further guidance on this problem, *Data access problems - DB2 database:*

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/
com.ibm.websphere.zseries.doc/info/zseries/ae/rtrb_dsaccess3.html
```

If you are still experiencing the problem, update the PROCs for the servant address spaces to include the DB2 libraries.

If your installation does not have SDSNEXIT, SDSNLOAD, and SDSNLOD2 in the linklist, then you must update the //STEPLIB DD concatenation for the servant address space with the missing libraries.

13.3.11 Data source exception: storage allocation error

DB2 for z/OS only

The symptom of this problem is that the messaging engine fails to start and these messages are logged to SystemOut as in Example 13-7.

Example 13-7 Data source exception: unavailable resource

```
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.readDataFromPersis
tence(AbstractItemLink.java:2487) at
com.ibm.ws.sib.msgstore.cache.links.AbstractItemLink._restoreItem(Abstr
actItemLink.java:639)
```

When using the DB2 Universal JDBC driver in type 2 mode, the driver can allocate excessive amounts of memory in order to hold result sets, exhausting the available storage space.

Solution

You can avoid this error by performing one of the following steps:

- Set the JDBC driver custom property fullyMaterializeLobData to false. In the administrative console:
  a. Select **Resources** → **JDBC** → **JDBC Providers**.
  b. Click the name of the JDBC provider you have configured.
  c. Select **Data sources**.
  d. Click the names of the data source you have configured.
  e. Select **Custom properties**.
  f. Click **fullyMaterializeLobData** and ensure the value is set to **false**.
- Or, use the Universal JDBC provider of type 4 driver.

Restart the application server after making your change.

13.4 Messaging engine not accepting work

A messaging engine will stop accepting work if the exclusive lock on its data store has been lost. The messaging engine will refuse any further work until it has successfully reacquired the lock.

If your messaging engine stops accepting work during normal run time, you will start to see rollback exceptions in your applications and application server log files.

13.4.1 Analyze SystemOut

Analyze SystemOut for the messaging engine. Look for any CWSIS*xxxxx* messages. In particular, pay attention to error messages, but information and warning messages are useful as well.

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 27, “The next step” on page 581. If you see these messages, see 13.4.2, “Active messaging engine loses the lock on the data store” on page 386:

- `SibMessage I [:] CWSIS1546I: The messaging engine, ME_UUID=82EC72EC0F3555B6, INC_UUID=743A743A0F35629F, has lost an existing lock or failed to gain an initial lock on the data store.`

- `SibMessage I [:] CWSIS1538I: The messaging engine, ME_UUID=82EC72EC0F3555B6, INC_UUID=743A743A0F35629F, is attempting to obtain an exclusive lock on the data store.`
- `SibMessage I [:] CWSIS1519E: Messaging engine testNode01.server1-default cannot obtain the lock on its data store, which ensures it has exclusive access to the data.`

13.4.2 Active messaging engine loses the lock on the data store

If the messaging engine stops accepting work during normal run time, you see messages like those in Example 13-8. These messages indicate that a messaging engine has lost the exclusive lock on the data store at run time. The messaging engine stops accepting work and will refuse any further work until it has successfully reacquired the lock. You might see rollback exceptions in the messaging application logs. If the messaging engine continues trying to get the lock, you see many repeating occurrences of these messages.

Example 13-8 Database lock lost during runtime

```
SibMessage I [:] CWSIS1546I: The messaging engine,
ME_UUID=82EC72EC0F3555B6, INC_UUID=743A743A0F35629F, has lost an
existing lock or failed to gain an initial lock on the data store.

SibMessage I [:] CWSIS1538I: The messaging engine,
ME_UUID=82EC72EC0F3555B6, INC_UUID=743A743A0F35629F, is attempting to
obtain an exclusive lock on the data store.
```

In a failover configuration, the problem can be caused by a temporary glitch in the network that caused a failover messaging engine to start and to take a lock on the database. In this case, the original messaging engine should produce the message shown in Example 13-9.

Example 13-9 CWSIS1519 at failover

```
SibMessage I [:] CWSIS1519E: Messaging engine
testNode01.server1-default cannot obtain the lock on its data store,
which ensures it has exclusive access to the data.
```

Solution

This problem is most likely caused by connectivity problems to the database, such as a network or database server outage. Resolve these problems. If network connectivity and the database server are both functional, check the rest of the messaging engines in your failover cluster to see which is currently holding the lock on the database. The instance that is successfully holding the lock should have started and be accepting work.

13.5 Validate solution

Restart the messaging engine or bus member. Check the messaging engine state to ensure it has started and is accepting work.

14

File store problem determination

WebSphere Application Server Version 6.1 introduced a new persistence option, the *file store*, for the default messaging provider message store. This option uses flat files, on a local or remote file system, to store all persistent data used by the WebSphere Application Server default messaging system. File store is the default persistence mechanism for any new messaging engines you create.

This chapter addresses problems with messaging engines that use a file store for message persistence. If you are using data store, see Chapter 13, “Message store with data store persistence” on page 369. If you are not sure which mechanism you are using for persisting messages, see 12.1, “Determine message store type” on page 366.

14.1 Introduction to using file stores

Proper planning and configuration of the file store can prevent problems from occurring. This section discusses issues related to the use of file stores.

For a description of the file store and its configuration, see *File stores* at:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/
com.ibm.websphere.pmc.doc\tasks\tjm0002_.html
```

14.1.1 Symptoms of a file store problem

Symptoms of a problem with the messaging engine file store include:

- Messaging engine fails to start.
- Applications experience transaction rollbacks.
- The SystemOut log includes a java.io.FileNotFoundException message.
- The SystemOut log includes a com.ibm.ws.sib.msgstore.PersistenceException message.
- TheSystemOut log includes a com.ibm.wsspi.sib.core.exception.SIRollbackException message.

14.1.2 Sizing your file store files

There are three files which make up a working file store:

- Log: default size 100MB
- Permanent Store: default sizes min=200MB, max=500MB
- Temporary Store: default sizes min=200MB, max=500MB

In most cases running out of space in your file store files can be corrected by tuning the sizes of files in your system to match the requirements of your specific workload. The following sections discuss scenarios to consider when estimating the size of files you will need.

Send and receive large numbers of messages

If your workload consists of a high throughput of messages then overall storage capacity should not be a priority. Any message in the system is not likely to last long and so it might not even make it to disk before being consumed. In this kind of scenario you need to consider how many messages are likely to be on disk at any one point in time and to ensure that your files are big enough to manage. To To determine the correct values for your file sizes in this kind of scenario, you might get the best results by experimenting with the system under expected workloads.

Allow messages to build up for processing later (batch)

In some systems there might be a requirement to accept messages over a period of time without consuming them to be processed at the end of the day (for example, product stock updates). In these cases, the size of your file store files needs to match or exceed the storage requirements for the amount of message data you expect to build up.

For example, if you expect up to 1000 messages to be received, each with a size below 1 MB, you would need up to 1000 x 1 MB = 1000 MB of space in your store files. In practice, the size required is more than 1000 MB because space is reserved in your store files for system purposes. A good estimate for this extra space is the size of your log file. For example, if your log file is 100 MB you would estimate 1100 MB for your store file maximum size.

Factoring in some variance in your expected workload when you calculate your file sizes; overestimating to some extent might be a safer, more reliable approach. Because disk storage is relatively cheap it is better to have a working system with unused file space than to have a system come down due to lack of space. Also, if all of your messages use a persistent reliability, you only need to set your permanent store file size limits to this value. If you do not make use of the non-persistent reliabilities (or vice versa), there is no need to have your temporary store file this large.

Message buildup during a messaging engine failover

A messaging system that consists of producers and consumers that are on separate remote messaging engines requires some consideration when planning for failover.

For example, suppose you have three servers:

- server1: Producer application and Messaging Engine 1
- server2: Consumer application and Messaging Engine 2
- server3: Failover server for server2/Messaging Engine 2

Standard behavior in this scenario would see messages being routed from the producer application on server1 through Messaging Engine 1 to Messaging Engine 2 on server2 and then to the consumer application. If, however, server2 fails over to server3, the messages being produced by the application on server1 build up in Messaging Engine 1 waiting for Messaging Engine 2 to come up. In the period of time in which Messaging Engine 2 is in the process of failing over and the producing application is still running, a buildup occurs in Messaging

Engine 1; you need to consider this build up when you plan your file store file sizes.

To estimate the amount of extra space you need in your file store files in this scenario, consider the size and throughput of messages being created on server1.

For example, if server1 is producing 1000 messages a second at a size of 10 KB, your system is creating 1000 x 10 K x 60 = 600000 K message data each minute. So, for each minute that the failover of server2 takes, you have a build up of approximately 600 MB of data in server1. The normal production and consumption cycle of your file store does not need a large amount of file space (due to the short life of the messages). However, if you need to allow server2 three minutes to fail over, you must plan on your store files being able to hold more than 1800 MB of data. Again, you need to estimate this final value based on your expected failover time for server2, which you will determine through experimentation at development time.

14.1.3 File system requirements for file store

The file store relies heavily upon the capabilities of the underlying file system. In order to set up a reliable system, there are some very simple but important functions and guarantees that must be provided. In this section, we outline the basic requirements that a file store has on the file system it is using, and we give reasons for their importance. Because there are a large number of file system implementations available (NTFS, NFS, HPFS, SAN, and so on) we cannot discuss implementation specifics; however, by describing these requirements, we hope to provide you with the ability to determine if the file system you choose is suitable for your file store files.

Exclusive read/write file locking

To ensure exclusive access to the data stored in your file store files, your file systems need to support locking of files for read/write access. This requirement is especially important in high availability (HA) and clustering scenarios where your files might be accessed remotely, and the possibility that two servers might try to access them simultaneously. If the correct locking semantics are not supported by the file system hosting your file store files, two server processes could modify the data stored in them and, therefore, produce in an inconsistent state for both servers.

Disk force on write

When data is written to disk by the file store, it is written as part of the user transaction and it has a direct affect on the outcome of that transaction. To achieve the correct resolution of the user’s transaction, you need to be guaranteed at write time that the data that is written has made it to the physical disk.

For example, if the data is buffered in memory at write time, then the file store returns from the transaction successfully (commit) believing that the data is safe. If the buffered data is then lost because of a system failure, the system is inconsistent and the data for that transaction is lost.

This is very dangerous behavior which destroys the reliability of your messaging system. To avoid this situation, the file system used to support your file store files must ensure that when a java.nio.channels.FileChannel.force() is called, all waiting data is written to the physical disk (or, in the case of hardware-assisted file systems, at least guaranteed to make it to disk in the case of failure).

Accurate reporting of file space allocation

Another area that affects the reliability of your file store is accurate allocation of file system space at allocation time. If the file store allocates 100 MB of space on disk, it must be able to rely on the availability of the full 100 MB to accurately guarantee the success of writing data to disk.

For example, when a file store allocates 100 MB of file space it is expecting an exclusively allocated piece of disk that is 100 MB in size to be available to it. If the file system checks the free space on the disk successfully but does not allocate the space for writing solely by the file store, another process could use up some of the space that makes up the 100 MB allocated by the file store. If the file store tries to write to disk, expecting there to be plenty of space left in the 100 MB it allocated, but fails because of lack of space, not only has a failure occurred but the file store now has no accurate way of gauging how much free space is available for use.

A problem like this leads to an inconsistency between the disk and the file store which will result in the system being unable to fulfill new requests reliably.

Do not use file system level compression

Similar to the problem described above, using compression on your file system stops the file store from accurately measuring the amount of free space available in its files. Because the data written to disk is a compressed version of what the file store holds, the size of the data differs (at a variable rate), making any space calculations inaccurate.

14.2 Collect diagnostics

In addition to the SystemOut log files for the messaging engine and the application log collected in 10.3, “Collect diagnostics” on page 339, collect the first failure data capture (FFDC) logs, which are located in:

```
profile_root/logs/ffdc
```

Each log file name is prefixed with the originating server name.

The FFDC logs are primarily intended for use by IBM Service. However, you might find it useful to search this repository as part of your problem determination exercise.

14.3 Analyze diagnostics

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 27, “The next step” on page 581:

1. Inspect the SystemOut log for CWSIS0002E with a nested CWSOM1017E message as in Example 14-1.

Example 14-1 CWSOM1017: Unable to access file store location

```
SibMessage E [:] CWSIS0002E: The messaging engine encountered an
exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: Unexpected exception
caught starting persistent message store: CWSOM1017E:
ObjectManagerState=ObjectManagerState(X:\Log)/Stopped
ManagedObject(0/0)/Ready/26202620(ObjectManagerState) caught
exception=X:\Log(Exception) trying to locate or create log file
name={2}(String).
```

   This message indicates a messaging engine is not able to access the file system locations that have been provided for use by the file store files. Reasons why you could receive this message are:

   – Incorrect path. the locations provided do not exist.

   – User account not authorized. The user account under which WebSphere runs does not have authorization to read and write to the file system location.

2. To further narrow the appropriate action, check the application server FFDC log for an exception that matches the message seen in SystemOut:

   – **java.io.FileNotFoundException: *path* (The system cannot find the path specified.)**

     If you see this type of exception indicating the system could not find the path, note the path and see 14.4.1, "Incorrect path" on page 396.

   – **java.io.FileNotFoundException: *path* (Access is denied.)**

     If you see this type of exception indicating access was denied, see 14.4.2, "User account not authorized" on page 397.

3. Inspect the SystemOut log for CWSIS0002E with a nested CWSOM1042E message as in Example 14-2.

Example 14-2 Messaging engine exception: CWSOM1042E

```
SibMessage E [:] CWSIS0002E: The messaging engine encountered an
exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: Unexpected exception
caught starting persistent message store: CWSOM1042E:
ObjectStore=AbstractObjectStore(F:\\PermanentStore)/69946994(ObjectStor
e) was asked to allocate space for
ManagedObject=ManagedObject(null/null)/Constructed/22f422f4(ManagedObje
ct) when it was full.
```

   The CWSOM1042E message indicates a space constraint for the file store has prevented the messaging engine from starting.

   If you see this message, see 14.4.3, "File store has a space constraint during startup" on page 398.

4. Inspect the SystemOut log for CWSIS1002E with a nested message in the range of CWSIS1573E - CWSIS1576E. You might also see CWSM1042.

   These messages indicate that the file store experienced a space constraint during runtime. The CWSIS157xE message provides more detailed information about the file that failed as in Example 14-3.

Example 14-3 CWSIS157xE

```
SibMessage E [:] CWSIS1573E: The file store's log file is full.
SibMessage E [:] CWSIS1574E: The file store's permanent store file is
full.
SibMessage E [:] CWSIS1575E: The file store's temporary store file is
full.
```

   If you see CWSIS1002E and any of these messages, see 14.4.4, "File store has a space constraint during runtime" on page 398.

14.4 Causes and solutions

The initial barrier to a working message file store is the successful creation of the files that store all persistent data. The most common reasons a messaging engine is unable to create file store files at startup are:

- Incorrect path
- User account not authorized

Given that space in the file store files is finite (even when the store files are set to “unlimited” your disk space is finite), it is likely that you will run out of space when you first experiment with tuning the size of your files to meet your requirements. The symptom is the messaging engine fails to start or an application is unable to complete a transaction. The most common reasons are:

- File store has a space constraint during startup.
- File store has a space constraint during runtime.

14.4.1 Incorrect path

The primary symptom of a messaging engine unable to access the file store location is that the messaging engine fails to start and a CWSOM1017E message is logged to SystemOut. The corresponding entry in the FFDC log looks like Figure 14-4.

Example 14-4 PersistenceException: CWSOM1017E

```
Exception = com.ibm.ws.sib.msgstore.PersistenceException
Source = com.ibm.ws.sib.msgstore.impl.MessageStoreImpl.start
probeid = 755
Stack Dump = com.ibm.ws.sib.msgstore.PersistenceException: Unexpected
exception caught starting persistent message store: CWSOM1017E:
ObjectManagerState=ObjectManagerState(X:\Log)/Stopped
ManagedObject(0/0)/Ready/26202620(ObjectManagerState) caught
exception=X:\Log(Exception) trying to locate or create log file
name={2}(String).
at
com.ibm.ws.sib.msgstore.persistence.objectManager.PersistentMessageStor
eImpl.start
at com.ibm.ws.sib.msgstore.impl.MessageStoreImpl.start
at com.ibm.ws.sib.admin.impl.JsMessagingEngineImpl.start
at com.ibm.ws.sib.admin.impl.HAManagerMessagingEngineImpl.activate
at com.ibm.ws.sib.admin.impl.JsActivationThread.run
Caused by: com.ibm.ws.objectManager.NonExistentLogFileException:
CWSOM1017E: ObjectManagerState=ObjectManagerState(X:\Log)/Stopped
```

```
ManagedObject(0/0)/Ready/26202620(ObjectManagerState) caught
exception=X:\Log(Exception) trying to locate or create log file
name={2}(String).
at com.ibm.ws.objectManager.ObjectManagerState.<init>
at com.ibm.ws.objectManager.ObjectManager.initialise
at com.ibm.ws.objectManager.ObjectManager.<init>
at
com.ibm.ws.sib.msgstore.persistence.objectManager.PersistentMessageStor
eImpl.start
... 4 more
Caused by: java.io.FileNotFoundException: X:\Log (The system cannot
find the path specified.)
```

The root cause of this exception is in the message for the FileNotFoundException: the system cannot find the path specified.

Solution

Review the values you have set in the file store configuration to ensure that they refer to valid file system locations.

14.4.2 User account not authorized

This problem is characterized by the following FFDC entry:

Example 14-5 FileNotFoundException: Access is denied

```
Caused by: java.io.FileNotFoundException: X:\Log (Access is denied.)
at java.io.RandomAccessFile.<init>
at java.io.RandomAccessFile.<init>
at com.ibm.ws.objectManager.ObjectManagerState.<init>
```

The root cause of this exception is in the message for the FileNotFoundException. The messaging engine is unable to create the necessary files because of security restrictions on the file system locations provided.

Solution

Ensure that the user account under which your WebSphere installation is running has read/write access to the file system you have specified in the file store configuration.

14.4.3 File store has a space constraint during startup

If the file system you are using to store your file store files does not have the space required to create or grow the files as defined by the file store configuration parameters, then your messaging engine can fail to start.

The symptoms of this problem are that the messaging engine does not start and message CWSIS0002 and a nested CWSOM1042 are logged to SystemOut.

Example 14-6 Messaging engine exception: CWSOM1042E

```
SibMessage E [:] CWSIS0002E: The messaging engine encountered an
exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: Unexpected exception
caught starting persistent message store: CWSOM1042E:
ObjectStore=AbstractObjectStore(F:\\PermanentStore)/69946994(ObjectStor
e) was asked to allocate space for
ManagedObject=ManagedObject(null/null)/Constructed/22f422f4(ManagedObje
ct) when it was full.
```

In this case, a lack of space was first noticed when trying to create the PermanentStore file; however, this problem could equally apply to the TemporaryStore or the log file.

Solution

Review the file store configuration to ensure there is enough disk space available at the file store location to store at least the minimum expected size of the files.

14.4.4 File store has a space constraint during runtime

If the file store system faces a space constraint, because of either disk space requirements or configured size limitations, you will experience problems with messaging applications during run time.

The symptoms of this problem include message CWSIS1002 with nested message in the range (CWSIS1573 to CWSIS1576) in the SystemOut log for the messaging engine.

The first sign of reaching the limit of one of your file store files will most likely be experienced by a messaging application in your WebSphere environment. When the application attempts to commit a transaction containing some messaging work, the transaction will roll back because of a lack of space in a file.

The following exception stack is an example for such a problem. This exception is seen by the user application and it is also written to the server's FFDC directory.

Example 14-7 SIRollbackException: CWSIS1002E

```
com.ibm.wsspi.sib.core.exception.SIRollbackException: CWSIS1002E: An
unexpected exception was caught during transaction completion.
at
com.ibm.ws.sib.msgstore.transactions.MSDelegatingLocalTransaction.commi
t
at com.ibm.ws.sib.msgstore.test.cache.RegisterMessage.testRegister
at com.ibm.js.test.LoggingTestCase.runTest
Caused by: com.ibm.ws.sib.msgstore.PersistenceFullException:
CWSIS1575E: The file store's temporary store file is full.
at
com.ibm.ws.sib.msgstore.persistence.objectManager.BatchingContextImpl.i
nsert
at com.ibm.ws.sib.msgstore.task.AddTask.persist
at
com.ibm.ws.sib.msgstore.persistence.objectManager.PersistentMessageStor
eImpl.commit
at
com.ibm.ws.sib.msgstore.transactions.MSDelegatingLocalTransaction.commi
t
Caused by: com.ibm.ws.objectManager.ObjectStoreFullException:
CWSOM1042E:
ObjectStore=AbstractObjectStore(X:\Build\SIB\ws\code\sib.msgstore.impl\
build/filestore\TemporaryStore)/3eb83eb8(ObjectStore) was asked to
allocate space for
ManagedObject=ManagedObject(null/null)/Constructed/2ca62ca6(Persistable
SlicedData[ BINARYDATA ])(ManagedObject) when it was full.
```

In the case above, the application transaction was rolled back because the store file that is needed to store the message on disk was full. In this example, the store file is clearly stated as the temporary store, which means that it was an ExpressNonPersistent or ReliableNonPersistent message being committed at the time.

When either of these asynchronous reliability levels is used, you might see another message that can alert you to a lack of space in your file store files:

```
SibMessage E [:] CWSIS1576E: The dispatcher is full.
```

Solution

When you see that your system is constrained by a lack of free space in any of its file store files, review the current size limits to determine if they are suitable for your expected workload.

Check that there is sufficient space on the disk for your store files to grow to the upper limit you have set in the configuration.

14.4.5 Validate solution

If the problem prevented the messaging engine from starting, restart the messaging engine, server or cluster and verify that it starts. Check that the error message is no longer being generated and written to the SystemOut logs.

If you were experiencing a space problem with the file stores, restart the messaging engine and the messaging applications. Check that the error messages are no longer being generated and written to the SystemOut logs.

15

JMS application problem determination

This chapter describes how to diagnose and resolve problems that occur with JMS applications that use the default messaging provider. Problems include the inability to connect to a service integration bus and problems with sending (producing) or receiving (consuming) messages.

For more information about connecting applications to a service integration bus see *Connecting applications to a service integration bus* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001_.html

15.1 Symptoms of a JMS application problem

Symptoms addressed in this chapter include:

- A JMS application is unable to create a connection to a service integration bus.
- A JMS application is unable to produce messages to the WebSphere Application Server default messaging provider
- JMS application unable to consume messages from the WebSphere Application Server default messaging provider.
- You see `javax.jms.JMSException` messages in the application or SystemOut logs.
- You see `com.ibm.ws.sib.comms.server.ObjectStoreFullException` messages in the application or SystemOut logs.
- Messages are lost.

15.2 Analyze diagnostics

The primary symptoms of a JMS application problem are found in the application messages. You might also find additional information in the SystemOut log for the messaging engine involved in the connection attempt.

15.2.1 Verify the messaging engine is started

If you expected the client application to connect to a specific messaging engine, first check that the messaging engine and the server on which it is running are started.

To determine the messaging engine to which the application is attempting to connect, see 10.2.2, “Determine the messaging engine connected to an application” on page 327.

To determine the state of a messaging engine, see 10.2.3, “Determine your messaging engine status” on page 328.

If the messaging engine is not started and will not start, you need to diagnose the cause of that problem first. See Chapter 11, “Messaging engine problem determination” on page 355.

15.2.2 Analyze application exception messages

Check the application log for an exception stack trace.

Search for **`javax.jms.JMSException`** and determine what the application was attempting to do when it received this exception. Start with the `JMSException` and note the messages for each of the linked exceptions until you reach the bottom. Then, you will understand the path the problem took through the system code:

1. If the exception occurred on a `createConnection`, `createQueueConnection`, or `createTopicConnection`:
   - If your client is *not* running on an application server, see 15.3, “JMS application outside the server container cannot connect to bus” on page 404.
   - If your client *is* running on an application server, see 15.4, “JMS application in server container cannot connect to the bus” on page 412.
2. If the exception occurred on a `createProducer`, `createQueueSender`, or `createTopicPublisher`, or when the application attempted to produce a message (calling producer.send), see 15.5, “JMS application unable to produce messages” on page 417.
3. If the exception occurred when the application attempted to create a consumer for a given destination by calling `createConsumer`, `createQueueReceiver`, or `createTopicSubscriber`, see 15.7, “JMS application unable to consume messages” on page 423.
4. If the exception occurred when the application attempted to consume a message by calling `consumer.receive`, or if the application was running in the client container and an exception was received on the Connection `ExceptionListener` when using a `MessageListener`, see 15.7, “JMS application unable to consume messages” on page 423.
5. If you do not see errors, and the application seems to have successfully sent the message but it never reaches the expected destination, see 15.6, “Message produced but does not arrive at destination” on page 422.
6. If you do not see errors, and the application seems to have successfully created a consumer for the destination but does not receive any messages (that is, calls to `receive` return `null`, or a `MessageListener` is not invoked) see 15.7, “JMS application unable to consume messages” on page 423.

15.3 JMS application outside the server container cannot connect to bus

When a JMS application runs outside of the application server (for example, in the J2EE client container or as a thin client in a J2SE™ environment) and the application calls the `createConnection` method to create a connection to the service integration bus , the following steps take place. (These steps are transparent to the application.):

1. A connection is created to one of the application servers in the cell that hosts the appropriate service integration bus. The server that is used is called the *bootstrap server,* and it must have the SIB Service enabled on it.
2. After the connection to a bootstrap server is established, the client application can query the bootstrap server to determine where to go to connect to a messaging engine on the specified bus. This query might also include additional constraints (called *target properties*) specified by the user on the JMS ConnectionFactory. These constraints limit the set of messaging engines on the bus to which the application will be allowed to connect ; for example, they might specify connecting to any messaging engine inside a specified cluster bus member.
3. When a suitable target messaging engine has been found by the query, the messaging engine might be on a different server than the bootstrap server in which the query was executed. In this case, the application is redirected by the bootstrap server; it disconnects from the bootstrap server and creates a new connection to the server that hosts the target messaging engine. If the target messaging engine is hosted by the bootstrap server, the original connection is used instead of unnecessarily closing it and creating a new one.

To diagnosis a failure to connect to the bus, determine if:

1. The application was able to contact a bootstrap server.
2. The bootstrap server query successfully located a target messaging engine that matched the user specified target properties.
3. A redirect was necessary (the target messaging engine was on a different server) and if the redirect was successful.

15.3.1 Symptoms of a connection problem

The symptom of this type of problem is most apparent in the application messages.

The J2EE client application throws a `java.jms.JMSException` when attempting to connect to the bus. The `JMSException` is thrown by one of the following:

- `createConnection`
- `createQueueConnection`
- `createTopicConnection`

Error messages following the exception indicate the problem.

15.3.2 Analyze the exception in the application log

To diagnose the problem, first examine the `JMSException` and its associated linked exceptions that were caught by the application. See the listing in Example 15-1 on page 406.

Search for the following messages:

- **`javax.jms.JMSException: CWSIA0241E`**`: An exception was received during the call to the method JmsManagedConnectionFactoryImpl.createConnection: com.ibm.websphere.sib.exception.SIResourceException`

  Note the full content of the message and continue searching the log for error messages.

- If you see any of these messages, see 15.3.3, "Unable to contact the bootstrap server" on page 408:
  - **`CWSIT0006E:`** `It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to: {0}`
  - **`CWSIT0007W:`** `It is not possible to contact the bootstrap server at {0} because of exception: {1}.`
  - **`CWSIJ0063E:`** `A network connection to host name {0}, port {1} cannot be established.`
- If you see this message, see "Server does not permit client applications to connect" on page 409:
  - **`CWSIT0090E:`** `A bootstrap request was made to bus {0} using channel chain {1}. Use of this chain is not permitted by bus {0}.`
- If you see any these messages, see 15.3.4, "No messaging engine matches the target properties" on page 410:
  - **`CWSIT0019E:`** `No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}`

  - **CWSIT0088E:** `There are currently no messaging engines in bus {0} running. Additional failure information: {1}`
  - **CWSIT0102E:** `A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.`

    CWSIT0102E would be included as a cause of CWSIT0019E or CWSIT0088E (in the {2} or {1} insert position respectively).

- If you see this message, see 15.3.5, "Redirect to messaging engine fails" on page 411:
  - **CWSIT0092E:** `An attempt was made to connect to bus {0} using channel chain {1}. Use of this chain is not permitted by bus {0}.`

Example

A `JMSException` will resemble Example 15-1.

Example 15-1 JMSException CWSIA0241E

```
Caused by: javax.jms.JMSException: CWSIA0241E: An exception was received during the
call to the method JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It was not possible
to contact any of the specified bootstrap servers. Please see the linked exception
for further details. Bootstrap connections were attempted to:
[localhost:7289:BootstrapSecureMessaging].
   at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(JmsManag
edConnectionFactoryImpl.java:243)
   at
com.ibm.ws.sib.api.jms.impl.JmsQueueConnectionFactoryImpl.createQueueConnection(JmsQu
eueConnectionFactoryImpl.java:149)
   at
com.ibm.ws.sib.tools.explorer.helper.JMSHelper.getConnectionToME(JMSHelper.java:588)
   ... 3 more
Caused by: com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It was
not possible to contact any of the specified bootstrap servers. Please see the linked
exception for further details. Bootstrap connections were attempted to:
[localhost:7289:BootstrapSecureMessaging]
   at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.generateException(TrmSICoreC
onnectionFactoryImpl.java:928)
   at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.remoteBootstrap(TrmSICoreCon
nectionFactoryImpl.java:637)
```

```
   at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.createConnection(TrmSICoreCo
nnectionFactoryImpl.java:298)
   at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.createConnection(TrmSICoreCo
nnectionFactoryImpl.java:213)
   at
com.ibm.ws.sib.api.jmsra.impl.JmsJcaConnectionFactoryImpl.createCoreConnection(JmsJca
ConnectionFactoryImpl.java:672)
   at
com.ibm.ws.sib.api.jmsra.impl.JmsJcaConnectionFactoryImpl.createConnection(JmsJcaConn
ectionFactoryImpl.java:559)
   at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(JmsManag
edConnectionFactoryImpl.java:212)
   ... 5 more
Caused by: com.ibm.websphere.sib.exception.SIResourceException: CWSIT0007W: It is not
possible to contact the bootstrap server at localhost:7289:BootstrapSecureMessaging
because of exception: com.ibm.websphere.sib.exception.SIResourceException:
CWSIC1001E: A client attempted to connect with a remote messaging engine
(localhost:7289 - BootstrapSecureMessaging) 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 127.0.0.1, port 7289 cannot be established...
   at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.contactBootstrapService(TrmS
ICoreConnectionFactoryImpl.java:752)
   at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.remoteBootstrap(TrmSICoreCon
nectionFactoryImpl.java:580)
   ... 10 more
Caused by: com.ibm.websphere.sib.exception.SIResourceException: CWSIC1001E: A client
attempted to connect with a remote messaging engine (localhost:7289 -
BootstrapSecureMessaging) 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 127.0.0.1, port 7289 cannot be established..
   at
com.ibm.ws.sib.comms.client.ClientSideConnection.connect(ClientSideConnection.java:35
0)
   at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.contactBootstrapService(TrmS
ICoreConnectionFactoryImpl.java:682)
   ... 11 more
```

```
Caused by: com.ibm.ws.sib.jfapchannel.JFapConnectFailedException: CWSIJ0063E: A
network connection to host name 127.0.0.1, port 7289 cannot be established.
   at
com.ibm.ws.sib.jfapchannel.impl.octracker.ConnectionDataGroup.connect(ConnectionDataG
roup.java:480)
:
```

In this example, the application could not contact the bootstrap server (CWSIT0006E) because it could not open a socket to the bootstrap server at the specified host and port (CWSIJ0063E).

15.3.3 Unable to contact the bootstrap server

The symptoms of this problem are:

- **`CWSIT0006E:`** `It was not possible to contact any of the specified bootstrap servers. Please see the linked exception for further details. Bootstrap connections were attempted to: {0}`
- **`CWSIT0007W:`** `It is not possible to contact the bootstrap server at {0} because of exception: {1}.`
- **`CWSIJ0063E:`** `A network connection to host name {0}, port {1} cannot be established.`

If the connection attempt failed because it was unable to contact the bootstrap server, you must resolve the problem that caused the failure. In most cases this problem is caused by one of the following situations.

Provider endpoints in the connection factory were not set

The provider endpoints field of the ConnectionFactory that defines where the bootstrap servers are located has not been set. Therefore, the provider endpoint has been defaulted, but the defaulted value does not point to a server.

Solution

Configure the provider endpoints field to contain the *host:port:chain* of one or more bootstrap servers as described in the following Information Center topic *Configuring a connection to a non-default bootstrap server* at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/tasks/tjn0033_.html`

Bootstrap server is not started

The bootstrap servers are not started; therefore, a connection to the specified endpoint cannot be established.

Solution

Ensure that the application server that you are expecting the client application to connect is running.

Server does not permit client applications to connect

In some cases (particularly where security is enabled) the server might not permit client applications to connect using certain types of transport chain. For example, when bus security is enabled (the default) the use of the BootstrapBasicMessaging chain is not permitted.

The symptom for this problem is:

```
CWSIT0090E: A bootstrap request was made to bus {0} using channel chain
{1}. Use of this chain is not permitted by bus {0}.
```

Solution

Perform *one* of the following actions to resolve the problem:

- Modify the application connection factory to use a permitted chain (the preferred approach).
- Modify the server configuration to permit the use of the requested chain. For information about the use of permitted chains see the Information Center topic, *Transport security in service integration bus* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjr0490_.html

Messaging engine port conflict

If the messaging engine experienced a port conflict at startup, the application can experience the messages shown in Example 15-2.

Example 15-2 JMS client symptom of a port conflict

```
[:] CWSIT0007W: It is not possible to contact the bootstrap server at
localhost:7276:BootstrapBasicMessaging because of exception:
com.ibm.websphere.sib.exception.SIResourceException: CWSIC1001E: A
client attempted to connect with a remote messaging engine
(localhost:7276 - 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 127.0.0.1, port 7276 cannot be
established...
javax.jms.JMSException: CWSIA0241E: An exception was received during
the call to the method
JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It was
```

```
not possible to contact any of the specified bootstrap servers. Please
see the linked exception for further details. Bootstrap connections
were attempted to: [localhost:7276:BootstrapBasicMessaging].
```

This problem is due to a configuration error. If you see this message, review the ports assigned to the application server to ensure that there are no conflicts with the following ports:

- Service Integration Bus (SIB) port
- SIB secure port
- SIB MQ interoperability port
- SIB MQ interoperability secure port

If you suspect that there might be a port conflict, use a command such as **`netstat`** to view the ports that are currently in use.

15.3.4 No messaging engine matches the target properties

The symptoms of this problem are:

- **`CWSIT0019E:`** `No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}`
- **`CWSIT0102E:`** `A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.`
- **`CWSIT0088E:`** `There are currently no messaging engines in bus {0} running. Additional failure information: {1}`

CWSIT0102 would normally be seen as a cause of CWSIT0019 or CWSIT0088 (in the {2} or {1} insert positions, respectively).

If the application can contact to a bootstrap server, the process moves to the next step. The application queries the bootstrap server to locate a target messaging engine. These symptoms occur when it was not possible to locate a messaging engine that matched the target properties specified in the ConnectionFactory.

Solution

Check the target properties on the connection factory to ensure that they accurately describe the conditions that you wish to place on the connection. The properties to check include the BusName, Target, TargetType, TargetSignificance, TargetTransportChain and ConnectionProximity, and are quoted in the CWSIT0019E error message.

For information about the use of target properties see the Information Center topic, *Connecting applications to a service integration bus* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001_.html

ConnectionProximity property

A CWSIT0102E error message indicates a matching messaging engine has been ruled out due to the proximity constraint.

The ConnectionProximity property is taken relative to the bootstrap server. If the proximity is set to `Server` and the bootstrap server does not contain a messaging engine matching the target properties, a connection will never be created.

Messaging engine not running

If you are satisfied that the target properties accurately describe the set of messaging engines that you wish to have considered for connection, ensure that one or more of those messaging engines is running. You might observe the error message CWSIT0088E if none of the messaging engines are running.

For information about determining and changing the state of a messaging engine, see 10.2.3, “Determine your messaging engine status” on page 328.

15.3.5 Redirect to messaging engine fails

The symptom of this problem is this message:

```
CWSIT0092E: An attempt was made to connect to bus {0} using channel
chain {1}. Use of this chain is not permitted by bus {0}.
```

If the bootstrap server query successfully located a target messaging engine that matched the target properties, the application will have received instructions from the bootstrap server to close its current connection and redirect to another server (bus member) in the cell that is hosting the messaging engine that was chosen for connection.

A failure at this point is very rare, but it can happen if the information returned by the query has since become out of date; for example, the server hosting the messaging engine has crashed, or the messaging engine has failed over from that server to another server in the cluster at the same time as the client application was attempting to connect. Failures of this type are transient and can usually be resolved by waiting a short time and then retrying the connection attempt.

One example of a non-transient failure at this point is the case in which the targetTransportChain is not permitted by the target server, as exposed in the

CWSIT0092E error message. You can resolve this by changing the application connection factory to use a permitted chain (preferred), or by modifying the server configuration to permit the requested chain. For information about the use of permitted chains, see the Information Center topic *Transport security in service integration bus* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjr0490_.html

15.3.6 Validate the solution

The solution to your connection problem can be validated by re-running the client application so that it attempts to connect to the bus after you have made the necessary changes. The application should now connect successfully. If it does not do so, re-evaluate the exception information that is received by the application to determine whether the original problem has continued to occur, or whether a new problem is preventing the connection from being made.

In some circumstances (for example, modifying the chains permitted by the server), you might need to restart the application servers in order for changes to take effect.

15.4 JMS application in server container cannot connect to the bus

A JMS application running inside a server container is part of a J2EE artifact such as an EJB or servlet (that is, an application running in the EJB container or the Web container, respectively). If your problem relates to receiving messages using an MDB, refer to Chapter 18, “Message-driven beans problem determination” on page 465.

Applications running inside a server do not follow the same bootstrap process as those running outside the server container (described in 15.3, “JMS application outside the server container cannot connect to bus” on page 404) for contacting a bootstrap server, because they are already running in that environment. The logic used to find an appropriate messaging engine to connect to is the same for both types of clients after the bootstrap server has been contacted; the system uses the target properties specified in the application ConnectionFactory to determine the set of messaging engines that are active and acceptable for the conditions requested by the application. Failures to connect from inside the server container (or from the client environment after the bootstrap server has been successfully contacted) are usually due to the relevant messaging engines not being active, or because the target properties specified by the application do not match any of the messaging engines on the bus.

15.4.1 Symptoms of a connection problem

The symptom of this type of problem are most apparent in the application messages. The J2EE client application throws a `java.jms.JMSException` when attempting to connect to bus. The `JMSException` is thrown by one of the following:

- `createConnection`
- `createQueueConnection`
- `createTopicConnection`

Error messages following the exception indicate the problem.

15.4.2 Analyze the exception in the application log

To diagnose the problem, first examine the JMSException and its associated linked exceptions that were caught by the application. This examination will often be enough to tell you why the connection attempt was not successful.

Search for these messages:

- `JMSException` thrown by `createConnection`, `createQueueConnection`, or createTopic `connection`.

  ```
  javax.jms.JMSException: CWSIA0241E: An exception was received during
  the call to the method
  JmsManagedConnectionFactoryImpl.createConnection:
  com.ibm.websphere.sib.exception.SIResourceException
  ```

  Note the full content of the message and continue searching the log for error messages.

- If you see this message, see 15.4.4, “Bus name does not exist” on page 415.

  ```
  CWSIT0086E: Bus {0} not found
  ```

- If you see these messages, see 15.4.3, “No messaging engine matches the target properties” on page 414:

  – **CWSIT0019E:** `No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}`

  – **CWSIT0102E:** `A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.`

  `CWSIT0102E would normally be seen as a cause of CWSIT0019E.`

- If you see this message, see 15.4.5, “No messaging engines active” on page 415.

  CWSIT0088E: `There are currently no messaging engines in bus {0} running. Additional failure information: {1}`

- If you see this message, see 15.4.6, “Matching messaging engine found but not started” on page 416.

  CWSIT0104E: `The client attempted to connect to the messaging engine {0} on bus {1} but the connection could not be created because the messaging engine is not started.`

- If you see any of these messages, see 15.4.7, “Security authorization failure” on page 416:
  - **CWSIT0016E:** `The user ID {0} failed authentication in bus {1}.`

    `The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine.`
  - **CWSIT0022E:** `The user ID {0} failed authentication in bus {1}.`

    `The user ID used to create an intra-bus messaging engine connection failed to be authenticated by the remote messaging engine.`
  - **CWSIT0089E:** `The user ID {0} failed authentication in bus {1}.`

    `The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine.`

15.4.3 No messaging engine matches the target properties

The symptoms of this problem are these messages:

- **CWSIT0019E:** `No suitable messaging engine is available on bus {0} that matched the specified connection properties {1}. Reason for failure: {2}`
- **CWSIT0102E:** `A messaging engine selection {0} was found but had to be discarded because it does not satisfy the connection proximity constraint of {1} that was specified by the application.`

The problem is that one or more messaging engines are running but none match the specified target properties.

Solution

Check the target properties to ensure that they are correct, or start a messaging engine that matches the existing target properties. For example, a simple spelling mistake in the target field can prevent a connection from being made under any circumstances.

For information about the use of target properties see the Information Center topic, *Connecting applications to a service integration bus* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjb0001_.html

ConnectionProximity property

A CWSIT0102E error message indicates a matching messaging engine has been ruled out due to the proximity constraint.

When the application is running in a server environment, the ConnectionProximity property is taken relative to the server in which the application is running. CWSIT0102E occurs if the ConnectionProximity is set to `Server` and the local server does not contain a messaging engine.

15.4.4 Bus name does not exist

The symptom of this problem is this message:

```
CWSIT0086E: Bus {0} not found
```

The bus name that was specified does not exist in the configuration.

Solution

Modify the bus name property of the connection factory to refer to a bus that has been defined.

15.4.5 No messaging engines active

The symptom of this problem is this message:

```
CWSIT0088E: There are currently no messaging engines in bus {0}
running. Additional failure information: {1}
```

There are no messaging engines currently active in the specified bus.

Solution

Start one or more messaging engines in the bus.

15.4.6 Matching messaging engine found but not started

The symptom of this problem is this message:

CWSIT0104E: `The client attempted to connect to the messaging engine {0} on bus {1} but the connection could not be created because the messaging engine is not started.`

A matching messaging engine was found in the local process but it is not in started state.

Solution

Start the messaging engine and then retry the application.

15.4.7 Security authorization failure

The failure to connect is often caused by a security authorization or authentication constraint which might be revealed to the application as one of these messages.

Symptoms of an authorization problem include:

- **CWSIT0016E:** `The user ID {0} failed authentication in bus {1}. The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine.`
- **CWSIT0022E:** `The user ID {0} failed authentication in bus {1}. The user ID used to create an intra-bus messaging engine connection failed to be authenticated by the remote messaging engine.`
- **CWSIT0089E:** `The user ID {0} failed authentication in bus {1}. The user ID used to create a client connection with a messaging engine failed to be authenticated by the remote messaging engine`

This problem is usually due to an incorrect user ID or password specified for the connection to the remote messaging engine.

Solution

Ensure the user ID is authorized to connect to the bus.

See 25.5, "Authorization problems connecting to the bus" on page 559.

15.4.8 Validate the solution

Resolve the problem described in the exception received by the application and then retry the application connection attempt. In some circumstances (for example, where you have modified JNDI resources in order to fix the problem) you might need to restart the application servers for changes to take effect.

If the application does not successfully connect after you have made the appropriate changes, reevaluate the exception information that is now received by the application to determine whether the original problem has continued to occur, or whether a new problem is preventing the connection from being made.

15.5 JMS application unable to produce messages

This topic describes the problems that you might encounter when trying to produce, or send, a message from within a JMS application. It is assumed that the application has successfully connected to the bus.

15.5.1 Symptoms that an application is unable to produce messages

The symptoms of this type of problem are most apparent in the application messages.

- The JMS application throws a `java.jms.JMSException` when attempting to create a producer for a given destination. It throws the `JMSException` in one of the following methods:
  - `createProducer`
  - `createQueueSender`
  - `createTopicPublisherError`

  The messages following the exception indicate the problem.
- The JMS application throws a `java.jms.JMSException` when attempting to send a message (that is, calling `producer.send`).
- The application seems to have successfully sent the message but it never reaches the expected destination. No exception is received.

15.5.2 Analyze diagnostics

First, examine the `JMSException` and its associated linked exceptions that were caught by the application. This examination is often enough to tell you why the problem occurred.

Also, review the application server log files for the application process (server or client container) to see if there are any related entries around the time that the attempt was made to send the message.

Analyze the application log

Search for these messages:

- If you see this message, see 15.5.4, “Cannot attach to the requested destination” on page 419.

  `CWSIA0062E: Failed to create a MessageProducer for {0}`

- If you see this message, see 15.5.5, “Insufficient authorization to create a producer for the queue” on page 420.

  ```
  CWSIA0069E: The user does not have authorization to carry out this
  operation. See the linked exception for details.
  ```

- If you see this exception, see 15.5.6, “Application not closing producer objects” on page 420.

  `com.ibm.ws.sib.comms.server.ObjectStoreFullException`

- If you see any of this messages, see 15.5.7, “Current system state problems” on page 421.

  – `CWSIA0063E: Failed to send to {0}`

  – `CWSIK0025E: The destination {0} on messaging engine {1} is not available because the high limit for the number of messages for this destination has already been reached.`

  – `CWSIK0033E: The message reliability value {0} is greater than the destination reliability value {1} for destination {2} on messaging engine {3}.`

- Look for this message:

  ```
  CWSIA0067E: An exception was received during the call to the method
  {1}: {0}.
  ```

  Most other errors have CWSIA0067E as the top level message, and a detailed description of the problem is provided in the linked exceptions. In those situations, follow the problem diagnosis instructions for the type of error described in the linked exception.

If you do not see error messages, it is likely that the application successfully produced the message. If the message, however, does not arrive at the destination, see 15.6, “Message produced but does not arrive at destination” on page 422.

15.5.3 Causes and solutions

The most likely causes for failing to send a message are listed below along with references to the sections that describe how to resolve the problems.

Failures that occur while trying to create a JMS MessageProducer, QueueSender, or TopicPublisher include:

- CWSIA0062E - See 15.5.4, “Cannot attach to the requested destination” on page 419.
- CWSIA0069E - See 15.5.5, “Insufficient authorization to create a producer for the queue” on page 420.
- CWSIA0055E - See 15.5.6, “Application not closing producer objects” on page 420.

Failures that occur while trying to send a message (that is, calling `producer.send`) include:

- CWSIA0069E - See 15.5.5, “Insufficient authorization to create a producer for the queue” on page 420.
- CWSIA0063E - See 15.5.7, “Current system state problems” on page 421. These problems include:
  - CWSIK0025E - “Queue is full” on page 421
  - CWSIK0033E - “Destination maximum reliability lower than message reliability” on page 421.

If the application seems to have successfully sent the message but it never reaches the expected destination, see 15.6, “Message produced but does not arrive at destination” on page 422.

15.5.4 Cannot attach to the requested destination

The symptom of this problem is this message:

```
CWSIA0062E: Failed to create a MessageProducer for {0}
```

CWSIA0062E indicates that it was not possible to attach to the requested destination. The linked exception provides more details on the exact reason for the failure:

- This problem is usually because the requested destination has not been defined or an error was made in the JNDI destination objects.
- A variation on this problem is that the producer is being created for a temporary destination (usually to send a reply message) but the temporary

destination has been deleted, either because the originating application has explicitly asked for it to be deleted or because the originating application has disconnected, causing the temporary destination to be automatically deleted.

- In rare cases this problem might also indicate that the producer was expecting to connect to a queue and the destination is actually a topic space, or the other way around.

15.5.5 Insufficient authorization to create a producer for the queue

The symptom of this problem is this message:

```
CWSIA0069E: The user does not have authorization to carry out this
operation. See the linked exception for details.
```

CWSIA0069E indicates that the application does not have authorization to create a producer for the requested queue.

Solution

The linked exception provides further details.

If the message does not provide enough information to resolve the problem, see 25.6, “Authorization problems accessing a destination” on page 562 for further problem determination information.

15.5.6 Application not closing producer objects

The symptom of this problem is this message:

```
com.ibm.ws.sib.comms.server.ObjectStoreFullException
```

In long running or high turnover scenarios a badly behaved application might receive a linked exception containing a `com.ibm.ws.sib.comms.server.ObjectStoreFullException`. This exception indicates that the application has created thousands of producer objects without closing any of them. This exception is usually preceded by CWSIA0055E errors in the application server logs of the appropriate environment.

Solution

Modify the application so that it closes the MessageProducer object after it has finished using it.

15.5.7 Current system state problems

The symptoms of this problem include these messages:

- **CWSIA0063E:** `Failed to send to {0}`
- **CWSIK0025E:** `The destination {0} on messaging engine {1} is not available because the high limit for the number of messages for this destination has already been reached.`
- **CWSIK0033E:** `The message reliability value {0} is greater than the destination reliability value {1} for destination {2} on messaging engine {3}.`

Errors related to the current state of the system are reported as CWSIA0063E which contains a linked exception providing detailed information about what went wrong.

Queue is full

CWSIK0025E indicates that the destination is not available because it has reached the limit of messages that it has been configured to store; that is, the queue is full. The corrective action is to start the consuming application to cause messages to be consumed from the appropriate destination. After some messages have been consumed from the destination the sending application can send messages again.

Note: This message can occur for other reasons than the queue is full; for example, the queue is not found. The problem has been reported to IBM technical support.

Destination maximum reliability lower than message reliability

CWSIK0033E occurs when the service integration bus destination object has been configured with a maximum reliability which is lower than the reliability of the message that the producer is attempting to send.

15.5.8 Validate the solution

Resolve the problem described in the exception received by the application, and trigger the application so that it attempts to send another message. In some circumstances (for example, if you have made modifications to a JNDI resource such as a Queue object) you might need to restart the application servers so that changes take effect.

If the application does not successfully send the message after you have made the appropriate changes, re-evaluate the exception information that is now received by the application to determine whether the original problem has continued to occur, or whether a new problem is preventing the send from being successful.

15.6 Message produced but does not arrive at destination

In the event that the message was produced by the application without error but did not arrive at the destination, you can take the following actions to check for common problems.

Note: It is possible that an error is produced but either not caught and displayed by the application, or it occurred on a remote system, such as a foreign bus. Ensure that your application is catching and displaying all JMSExceptions.

15.6.1 Verify the transaction was committed

If the message was sent under a transaction, verify that the transaction has been committed. To determine if a message is still locked under a transaction, look at the runtime panel for the destination in question (for example the queue point) and view the available messages. A message marked `locked` is being processed under a transaction by another consumer.

If the transaction has not been committed, the message is not yet available for consumption by other applications. If the message is locked under a transaction, it has not been committed yet. If the transaction has been committed, the message is no longer visible on the queue.

In the server environment, control of the transaction is often handled by the container in which the application is running (for example, when using container-managed transactions). However, it might be under the control of the application (bean-managed transactions). In the client container or thin client environments, the transaction is always controlled by the application using the `session.commit` and `session.rollback` methods.

15.6.2 Verify the message was sent to the correct destination

It is surprisingly common for a message to be sent to a different destination than was expected.

Check which JNDI destination object is used by the application, and then check the contents of that destination object to ensure it matches your expectations.

15.6.3 Check for the message on the exception destination

Check the exception destination to see if your message has been moved there for some reason. If this is the case, the exception header will provide details about why it was moved there.

For information about determining if a the message is on an exception destination, see “Check the exception destination” on page 338.

15.6.4 Find the message

To determine how far through the system the message has traversed, see 10.2.4, “Finding queued messages” on page 329.

15.7 JMS application unable to consume messages

This topic describes some of the problems you might encounter when trying to receive a message using a JMS application. It is assumed that the application has successfully connected to the bus.

15.7.1 Symptoms that an application is unable to consume messages

The symptom of this type of problem will be most apparent in the application messages.

- The JMS application throws a `java.jms.JMSException` when attempting to create a consumer for a given destination. The `JMSException` is thrown by one of the following methods:
  - `createConsumer`
  - `createQueueReceiver`
  - `createTopicSubscriber`

  The messages following the exception indicate the problem.
- The JMS application throws a `java.jms.JMSException` when attempting to consume a message (that is, calling `consumer.receive`). Alternatively, if the application is running in the client container, an exception is received on the `ConnectionExceptionListener` when using a `MessageListener`.

- The application seems to have successfully created a consumer for the destination but does not receive any messages (that is, calls to `receive` return `null`, or a `MessageListener` is not invoked). No exception is received.

15.7.2 Verify integrity

If the JMS application has received a `JMSException`, the exception provides the primary source of diagnostic information for resolving this type of problem. In most cases, the cause is obvious when you examine the exception messages that caused the failure.

If the application did not receive an exception, but a call to `consumer.receive` returns `null`, there might still be a problem if you expect the message to be there.

Note: A call to `consumer.receive` returns `null` if there was no message available. The method returns successfully (with no exception) but you do not have a message to process. This result can be expected in some situations (for example, the producer might not have sent a message since the last time you checked). However, it could also indicate a problem.

Check the following possibilities first:

- Was the message successfully sent by the producing application?

  Look for a `JMSException` in the application log that indicates a problem producing messages (see 15.2, "Analyze diagnostics" on page 402). If these types of errors exist, address them first.

- Has the JMS connection been started?

  The JMS specification requires that messages not be delivered to consumers (synchronous or asynchronous) until the connection has been started. A CWSIA0087W warning message is returned to the application.

  If you see this message, modify your application to start the connection before calling receive.

- Did the message reach the destination from which the consumer is trying to receive the message?

  Messages can often be rerouted by business logic in mediations or administratively defined forward routing paths. Check that the producer and consumer are configured to produce and consume in the appropriate locations. In the case of queues or durable subscriptions, you can check the status by stopping the consuming application and using the runtime administration panel view of the queue point or subscription to view the available messages.

- If consuming from a queue, has the message already been consumed by another application or by a different part of the same application?

  This assessment usually requires a level of knowledge about what is going on in the system to identify. For example, the administrator would know whether other applications consume from the same destination.

  See 10.2.4, “Finding queued messages” on page 329 for help in tracking down missing messages.

  An advanced user might wish to turn on the SIB message trace to help with the identification process. To enable the trace using the administrative console, select **Troubleshooting** → **Logs and Traces** → ***server_name*** → **Change Log Details Levels.** On the Runtime tab, add `SIBMessageTrace=all=enabled` to the Groups, and click **Apply.**

- Has the message been locked by another application that is part way through processing it?

  To determine if this is the case, look at the runtime panel for the destination in question (for example the queue point) and view the available messages. A message marked `locked` is being processed under a transaction by another consumer.

- Check the exception destination to see if the message that was sent by the producer had a problem that meant it could not be sent to the target destination. In this case the message exception header provides details about what went wrong.

- The message also might not be available if it was sent with a reliability which caused it to be discarded (for example, if the message was sent as non-persistent and the server holding it was restarted). The message might have been discarded, even if the server was not restarted, if it was sent with a reliability of BestEffortNonPersistent and the system had constrained resources.

15.7.3 Analyze diagnostics

First, examine the `JMSException` and its associated linked exceptions that were caught by the application. This examination will often be enough to tell you why the attempt to consume a message was not successful.

Also, review the application server log files for the application process (server or client container) to see if there are any related entries around the time that the attempt was made to receive the message.

Analyze the application log

Search for the following messages:

- If you see this message, 15.7.4, "Cannot attach to the destination" on page 426.

  CWSIA0086E: `Failed to create a MessageConsumer for {0}`

- If you see this message, 15.7.5, "Not authorized for the requested queue" on page 427.

  CWSIA0090E: `The user does not have authorization to carry out this operation. See the linked exception for details.`

- If you see these messages, 15.7.6, "Application not closing consumer objects" on page 427.

  `com.ibm.ws.sib.comms.server.ObjectStoreFullException`

- If you see this message, examine the other associated messages.

  CWSIA0085E: `An exception was received during the call to the method {1}: {0}.`

  Most other errors generated when receiving messages have CWSIA0085 as the top level message, and detailed description of the problem is provided in the linked exceptions

15.7.4 Cannot attach to the destination

The symptom of this problem is this message:

CWSIA0086E: `Failed to create a MessageConsumer for {0}`

CWSIA0086E indicates that it was not possible to attach to the requested destination. The linked exception provides more details on the exact reason for the failure and indicates what needs to be done to resolve it.

This problem is usually because the requested destination has not been defined (or an error was made in the JNDI Destination object).

A variation on this problem occurs when the consumer is being created for a temporary destination (usually to consume a reply message) but the temporary destination has been deleted by the application. Alternatively, this application might not be permitted to consume from the temporary destination. The JMS specification requires that only the application that creates a temporary destination can consume from it.

In rare cases this problem might also indicate that the consumer was expecting to connect to a queue and the destination is a topic space, or the other way around.

15.7.5 Not authorized for the requested queue

The symptom of this problem is this message:

```
CWSIA0090E: The user does not have authorization to carry out this
operation. See the linked exception for details.
```

CWSIA0090E indicates that the application does not have authorization to create a consumer for the requested queue. The linked exception provides further details.

If the message does not provide enough information to resolve the problem, see 25.6, “Authorization problems accessing a destination” on page 562 for more problem determination information.

15.7.6 Application not closing consumer objects

The symptom of this problem is the following exception:

`com.ibm.ws.sib.comms.server.ObjectStoreFullException`

In long running or high turnover scenarios, a badly behaved application might receive a linked exception containing a `com.ibm.ws.sib.comms.server.ObjectStoreFullException`. This exception indicates that the application has created thousands of consumer objects without closing any of them. This is usually preceded by CWSIA0059E messages in the J2EE application log, client console or application server log, depending upon where the application is running. In this situation, modify the application to close the `MessageConsumer` object after it has finished using it.

15.7.7 Validate the solution

Resolve the problem described in the exception received by the application, and trigger the application attempt to consume the message again. In some circumstances (for example, when you have modified JNDI resources to resolve the problem), you might need to restart the application servers for changes to take effect.

If the application does not successfully send the message after you have made the appropriate changes, re-evaluate the exception information that is now received by the application to determine whether the original problem has continued to occur, or whether a new problem is preventing the send from being successful.

16

Messaging in a multiple messaging engine environment

Chapter 15, “JMS application problem determination” on page 401 discusses problems that prevent a JMS application from connecting to a service integration bus and problems that arise when an application attempts to send or receive messages using the default messaging provider.

This chapter covers JMS application problems that are specific to an environment that includes multiple messaging engines. In this type of topology, the producing and consuming applications might be connected to different messaging engines and, in the case of point-to-point messaging, might be connected to a messaging engine that is remote to the location of the message point. This chapter discusses problems specific to this type of messaging topology.

16.1 Symptoms of problems in a multiple messaging engine environment

The following symptoms are specific to networking in an environment with multiple messaging engines. If you experience any of these symptoms, see the referenced section:

- Messages are not found when expected on the queue.

  Messages have been produced to a destination by a messaging application but, the messages cannot be seen on the destination's message points or arrive after an unexpectedly long period of time.

  See 16.3, "Messages are lost or are slow to arrive" on page 431.
- Messages are not consumed from the queue when expected.

  A consuming application does not receive any messages even when the messages can be seen on the queue's queue points.

  See 16.4, "Messages are not consumed or consumed slowly" on page 433.
- Application fails with an SIMPLimitExceededException failure while sending messages to a queue.

  See 16.5, "SIMPLimitExceededException failure while sending messages to a queue" on page 434.
- Application fails with an SIMPLimitExceededException failure while sending messages to a topic space.

  See 16.6, "SIMPLimitExceededException failure while sending messages to a topic space" on page 436.

If these symptoms do not match your problem or if you are not sure, continue to 16.2, "Analyze diagnostics" on page 430 to collect and analyze log files.

16.2 Analyze diagnostics

Scan each log for indications of an error, as described below.

16.2.1 Analyze the application log

In the application log, look for JMSExceptions.

Specifically, scan for SIMPLimitExceededException. If you find this exception, see the appropriate section for your environment:

- 16.5, “SIMPLimitExceededException failure while sending messages to a queue” on page 434
- 16.6, “SIMPLimitExceededException failure while sending messages to a topic space” on page 436.

If you do not find this message but you find other JMSExceptions, see 10.6, “Guide to chapters by exception” on page 348.

16.2.2 Analyze SystemOut for the messaging engines

For messages to be transmitted between remote queue points and the local queue point (and hence messaging engines), the respective messaging engines must communicate with each other. The connectivity status between messaging engines within a bus is reported in the respective messaging engine’s application server log files.

In the messaging engine SystemOut log, scan for error messages with a prefix of CWSI:

- If you see CWSIT0029I, see 16.3, “Messages are lost or are slow to arrive” on page 431.
- If you see error messages other than this, check the list of messages in 10.5, “Guide to chapters by message” on page 342.

16.3 Messages are lost or are slow to arrive

If an application is connected to a different messaging engine than the one on which the identified message points are located, messages might be in transit between messaging engines.

For each possible messaging engine that the producer could have connected to, check the state of the remote queue point on the messaging engine.

Examine the message points for the messaging engine. If the messaging engine has been used to produce messages to the queue point in question, there will be a remote queue point, which identifies it (and the messaging engine hosting that queue point).

If the queue point is clustered, there might be multiple remote queue points.

For each remote queue point:

1. Check to see if there are any messages queued for transmission.

   Check the current outbound messages count on the remote queue point panel. If this value is greater than zero:

   a. Verify the messaging engine identified as hosting the message point is running. If it is not, start the messaging engine and check that the messages arrive.

   b. Check that the two messaging engines are able to communicate.

      Examine the application server log files and search for occurrences of the message code **CWSIT0029I**. If you find it, check the message body to see if the messaging engines match those hosting any queue points that interest you. For example:

      ```
      CWSIT0029I: The connection for messaging engine
      Messaging_Engine_1 in bus Bus_Name to messaging engine
      Messaging_Engine_2 stopped.
      ```

      This message generally implies some kind of network connectivity problem. Resolving connectivity issues enables the messages to be delivered to the queue point.

   c. Select the remote message point and then select outbound messages.

      Examine the state of the first message.

      If it is in the *committing* state, the transaction used to produce the message has not been successfully committed, which blocks this first message and subsequent messages from being transmitted. Resolve any problem with the associated transaction.

      If the state is *pending send*, the target messaging engine cannot be contacted. Check the application server log files to ensure that the messaging engine is running and that no communication errors have been reported.

2. If no messages are queued for transmission, but the number of `outbound messages sent` is greater than zero, then messages have been successfully sent from this messaging engine to the message point. Reconfirm that the message has not arrived and been removed from the queue point (for example, consumed by an application, expired, or moved to an exception destination).

16.4 Messages are not consumed or consumed slowly

When you think messages are being consumed slowly or not at all, your first step in diagnosing the problem is to find and examine the messages:

1. For each queue point containing messages, display the messages.

   If messages are available but are in a *locked* state, resolve why they are not available:

   – If a message is locked with a transaction ID, then the message has been deleted under a transaction that has not yet been committed.

   – If a message is locked with no transaction Id and stays in the locked state for a significant period of time, it is likely that it is in the process of being consumed by an application connected to another messaging engine.

   If no available messages were found, see 16.3, "Messages are lost or are slow to arrive" on page 431.

2. If messages are found and the destination consists of more than one queue point, determine if messages are found on all queue points or a subset of the queue points.

   A consumer will only consume from a single message point. A consumer might be attached to a message point that does not contain any messages. If a clustered queue is being used, you must ensure that all queue points have an active consumer to be sure that messages will be consumed.

   For more information about this behavior see *Workload sharing with queue destinations* at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc\concepts\cjt0014_.html

If an *application is connected to a messaging engine that owns one of the messaging points*, the application always consumes from only that queue point. If no messages are available on the queue point, then the application has no messages to consume. If messages are available, see 15.7, "JMS application unable to consume messages" on page 423 for general JMS consumer problem determination information.

If an *application is connected to a messaging engine that does not own a queue point*, the messaging engine attaches to any one of the remote queue points. If the application is running, you can see which queue points are attached by inspecting the remote queue points on the messaging engine where the application is connected:

1. Determine which messaging engine to which the application is connected.
2. Identify the remote queue points. If this messaging engine is currently, or was recently, being used to consume messages from the message point on a remote messaging engine, there will be a remote queue point for it. If the message point is clustered, there might be multiple remote message points.
3. Perform the following for each remote queue point:
   a. If the consuming application is currently waiting for a message, one of the remote message points has at least one current message request listed.

      If none are found, the consuming application is not connected to this messaging engine or is not actively consuming. Check the application for errors.

      Otherwise, if a message request is found, note the selector associated with the request. View the message point that corresponds to the remote message point to see if there are available messages that would match the selector. It is likely that there are no messages available on this particular message point suitable for the consuming application.

      If there are messages, proceed to the next step.

   b. If the consuming application is not currently waiting for a message or even connected it is harder to diagnose a possible reason for failing to consume messages.

      In this case, the completed message request property of a remote message point might be used to give an indication of which message point was being consumed from. Use this request property to inspect the message point to see if it contains any available messages. If it does not, it is likely that this is why the consumer did not receive any messages. Otherwise, if possible, re-run the consuming application and follow the previous steps.

16.5 SIMPLimitExceededException failure while sending messages to a queue

This section discusses steps to take to diagnose the problem when a producing application's send of a message to a *queue* fails with an

`SIMPLimitExceededException`. If the application is a JMS application, this exception will be wrapped by a `JMSException`.

16.5.1 Examine the queue point message depths

This error implies that the queue point (or remote queue point) for which the application is producing has exceeded its high message threshold.

To identify where the problem lies:

1. Determine the following information:
   - The messaging engine to which your application is connected.
   - The appropriate queue and, where applicable, any mediation points.
2. Check the current message depth of each queue point.

   If the current depth is at or above the high message threshold, the queue point is full. Reasons why this might occur are that:

   a. No consumer is actively consuming from this queue point.

      Ensure that each queue point has an active consumer.

   b. The active consumers cannot consume messages at a rate equal to the rate at which the producers are producing them. You can resolve this in one of two ways:

      - Increase the speed of the consumer applications; for example, increase the number of threads used by an MDB or connect consuming applications directly to this messaging engine.
      - Reduce the speed of the producing applications.

If one or more of the queue points has not reached the high message threshold, the failing producing application might be connected to a messaging engine that does not own the queue point for the destination. A remote queue point has been created by this messaging engine and the messages have been built up here:

1. Identify which messaging engines the producing application was or is connected to.
2. For each messaging engine, identify the remote queue points.
3. Check the current outbound message value of the remote queue point. If this value is high or it has reached the high message threshold of the messaging engine, check the following areas:

   a. Determine if the `outbound messages sent` value is increasing over time. Refresh the panel after a few seconds to see if the number changes:

i. If the `outbound messages sent` value is *not increasing*, make sure the messaging engine that owns the queue point is running. If not, start it. If it fails to start, see Chapter 11, “Messaging engine problem determination” on page 355.

   After the messaging engine is started, the outbound messages should drain from the remote queue point into the queue point. If the messages are not sent, examine the application server log files and search for occurrences of message CWSIT0029I. If you find this message, check the message body to see if the messaging engines match those hosting any queue points that interest you.

   For example:

   ```
   CWSIT0029I: The connection for messaging engine
   Messaging_Engine_1 in bus Bus_Name to messaging engine
   Messaging_Engine_2 stopped.
   ```

   This message generally implies a network connectivity problem. Resolving connectivity issues enables the messages to be delivered to the queue point.

ii. If the `outbound messages sent` value is *increasing*, messages are successfully being transmitted to the corresponding queue point. The reason for the large number of outbound messages is that the rate of message production is faster than they can be transmitted. This situation might be due to a slow network connection between messaging engines or to other network traffic interfering with this connection’s performance. Improving the network connection should increase message throughput. If it does not, the message production rate must be reduced to prevent the backlog.

16.6 SIMPLimitExceededException failure while sending messages to a topic space

This section discusses steps to take to diagnose the problem when a producing application’s send of a message to a *topic space* fails with an `SIMPLimitExceededException`. If the application is a JMS application, this exception will be wrapped by a `JMSException`.

16.6.1 Examine the publication point message depths

This error implies that the publication point for which that the application is producing has exceeded its high message threshold. This situation occurs if you have one or more subscriptions with a large number of messages queued on it.

These subscriptions are either application defined subscriptions or system level subscriptions, made by publication points on neighboring messaging engines.

Application subscriptions build up messages if no application is currently consuming messages from them, or if their consumption rate is slower than the message production rate.

Neighboring publication point subscriptions might build up messages if the neighboring messaging engine is not running, there is a problem communicating with it, or the message production rate is too fast for the network connection.

Neighboring publication points are represented on a local messaging engine as a remote publication point. The remote publication points transmit messages reliably from publishing applications connected to one messaging engine to subscriptions created on different messaging engines.

To diagnose which of these is causing the problem:

1. Determine the topic space for which the application is producing messages.
2. Identify the messaging engine to which the producing application is connected.
3. Display the publication point for the topic space on the connected messaging engine.
4. Select **Subscriptions** to list all current subscriptions.
5. Select each subscription to display its current message depth.

   A subscription with a number approaching the high message threshold of the publication point might be causing further production of messages to fail. If you find that this is the case, determine the reason for the high number of messages. The reason for this result is most likely one of these conditions:

   – No application is actively consuming the messages.

     Start an application or delete the subscription (if it is a durable subscription).

   – The consuming application cannot consume at the same rate as the producing applications are producing messages.

     Reduce the message rate of the producer.

6. If no local subscriptions have a backlog of messages, check for any remote publication points on the messaging engine.
7. Check the current `outbound messages sent` count for each remote publication point. If it is high or has reached the high message threshold of the messaging engine, check the following conditions:

a. If the `outbound messages sent` value is *not increasing*, make sure the messaging engine that owns the publication point is running. If not, start it. If it fails to start, see Chapter 11, “Messaging engine problem determination” on page 355.

   After the messaging engine is started, the outbound messages should drain from the remote publication point into the publication point and subscriptions. If the messages are not sent, examine the application server log files and search for occurrences of message CWSIT0029I. If you find this message, check the message body to see if the messaging engines match those hosting any queue points that interest you.

   For example:

   ```
   CWSIT0029I: The connection for messaging engine
   Messaging_Engine_1 in bus Bus_Name to messaging engine
   Messaging_Engine_2 stopped.
   ```

   This message generally implies a network connectivity problem. Resolving connectivity issues enables the messages to be delivered to the queue point.

b. If the `outbound messages sent` value is *increasing*, messages are successfully being transmitted to the corresponding publication point. The reason for the large number of outbound messages is that the rate of message production is faster than they can be transmitted. This condition can be due to a slow network connection between messaging engines or to other network traffic interfering with this connection’s performance. Improving the network connection should increase message throughput. If it does not, the message production rate must be reduced to prevent the backlog.

17

Clustering problem determination

This chapter walks you through the process of debugging WebSphere Application Server cluster problems associated with the default messaging provider.

17.1 Introduction to clustering for messaging

When configuring a clustered environment for messaging using the default messaging provider, an installation can follow one of these two approaches:

- Configure and use a cluster in a *highly available (HA) configuration*. In this configuration, only one messaging engine is active on one of the cluster members at any given time.
- Configure and use a cluster in a *workload management (WLM) configuration*. In this configuration, a number of active messaging engines equal to the number of cluster members are active simultaneously, each specifically pinned to a cluster member.

The decision for the style of the messaging cluster is made at configuration time, after the WebSphere cluster and cluster members have been created.

Note: A topic space cannot be workload-balanced in a WLM cluster.

For more information about messaging engine clustering, see:

- *Configure a Service Integration Bus in a network deployment environment*

  `http://www.ibm.com/developerworks/webservices/library/ws-sibus/`
- *Configure platform messaging and Web Services Gateway for a clustered environment, Part 1: Cluster enhancements for WebSphere Application Server Version 6.1 on z/OS*

  `http://www.ibm.com/developerworks/webservices/library/ws-clusterenv1/index.html`
- *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304
- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392

17.1.1 Symptoms of problems specific to clustering

The problems involved in the use of a clustering for the default messaging provider fall into two categories: configuration problems and runtime problems:

- The primary symptom of a configuration problem is that one or more messaging engines fail to start.

- The primary symptom of a runtime problem is the non-delivery or orphaning of messages. Another symptom is messaging engines starting in the wrong location.

Note: Some clustering problems might have symptoms of data store errors, specifically message CWSIS1546I. This problem can occur when two instances of a messaging engine attempt to connect to the same data store; the first successfully obtains the lock, and the second receives multiple CWSIS errors. This behavior is intentional. The data store prevents your messaging engine from starting in two places; so the data store prevents the potential result of corrupted messaging data. For more information about data store locking see Chapter 13, "Message store with data store persistence" on page 369.

17.2 Analyze diagnostics

In an HA configuration, although only one messaging engine is active at a time, it can run on any cluster member. You might need to look at the log for each cluster member to determine where that messaging engine is located.

In a WLM configuration, there is one active messaging engine per cluster member. You need to look at the SystemOut log for each member to determine the reason for a specific messaging engine not starting.

Analyze each SystemOut log to determine where the messaging engine attempted to start and if it was successful.

For each of the cluster members, view the SystemOut log until you find the log entries with the messaging engine. Look for messages that indicate a messaging engine has attempted to start and failed. In a WLM configuration, one messaging engine per cluster member should be started.

Search for messages with the following format:

- CWSID*xxxx*I
- CWSID*xxxx*E
- CWSIS*xxxx*E

17.2.1 Determine if the messaging engine started normally

The log for a messaging engine normal start will look similar to Example 17-1 on page 442:

Example 17-1 Messaging engine start up with file store

```
WsServerImpl  A   WSVR0001I: Server server1 open for e-business
:
SibMessage    I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Joined.
SibMessage    I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Starting.
:
SibMessage    I   [:] CWSIS1566I: A single previous owner was found in
the messaging engine's file store, ME_UUID=1D842649652FA294,
INC_UUID=0E040E040F8E07CD
SibMessage    I   [:] CWSIS1563I: The messaging engine,
ME_UUID=1D842649652FA294, INC_UUID=0D800D800FA453BF, has acquired an
exclusive lock on the file store.
:
SibMessage    I   [:] CWSID0016I: Messaging engine
WASNode01.server1-Bus1 is in state Started.
```

Note: If the messaging engines appear to have started normally on all applicable cluster members, see “Message delivery problems” on page 443.

A start up that failed will have error messages of the format CWSISxxxxE and CWSIDxxxxE. The key message is:

```
CWSIS0002E: The messaging engine encountered an exception while
starting. Exception: {0}
```

The exception information provides a reason why the messaging engine has failed to start.

Example 17-2 shows an example set of these types of messages.

Example 17-2 SystemOut log with messaging engine error

```
[…]CWSID0016I: Messaging engine ClusterHA.000-Bus1 is in state
Starting.

[…]CWSIS0002E: The messaging engine encountered an exception while
starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: CWSIS1524E: Data
source, jdbc/HA, not found.
```

```
[…]CWSID0035E: Messaging engine ClusterHA.000-Bus1 cannot be started;
detected error reported during
com.ibm.ws.sib.msgstore.impl.MessageStoreImpl start()

[…]CWSID0027E: Messaging engine ClusterHA.000-Bus1 cannot be restarted
because a serious error has been reported.

[…]CWSID0016I: Messaging engine ClusterHA.000-Bus1 is in state Stopped.

[…]CWSID0016I: Messaging engine ClusterHA.000-Bus1 is in state Joined.
```

Note: If the messaging engine did not start properly, see “Messaging engine fails to start” on page 444.

Message delivery problems

If the messaging engines appear to have started normally but you are having issues with message delivery or with the location of the messaging engine, look for the following symptoms.

Common problems that can occur in an HA or WLM configuration are:

- The messaging engine is not starting on the preferred server. In a WLM configuration, multiple messaging engines are started on one cluster member while no messaging engines are started on others. If either of these conditions exist, see 17.4.1, “Core group policy is incorrect or no preferred server defined” on page 454.
- Response messages appear on a different partition than the request message (messages appear in an unexpected place). If this appears to be your problem, see 17.4.2, “A permanent reply queue is in use” on page 458.
- Orphan messages after a failover. The failover might have been successful or not. See 17.4.4, “Orphaned messages after failover” on page 459.

Common problems that are specific to a WLM configuration are:

- Messages are not consumed from a partitioned destination. See 17.4.6, “Orphan messages on a partitioned destination” on page 462.
- Messages do not arrive on a specified partition of a partitioned destination. See 17.4.7, “Cluster weights incorrectly specified” on page 463.

If none of these symptoms fits your problem, see Chapter 10, “Introduction to JMS application problem determination” on page 321 to see if any other problem types might apply, or see Chapter 27, “The next step” on page 581 to gather documentation and contact IBM support.

Messaging engine fails to start

The failure of a messaging engine to start usually indicates a configuration problem.

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 27, "The next step" on page 581:

- If you find any of the following exception information, see 17.3.1, "JDBC provider or data source configuration error" on page 445.

  ```
  CWSIS1524E: Data source, data_source, not found
  ```

  ```
  CWSIS1501E: The data source has produced an unexpected exception:
  java.sql.SQLException: Database 'database_name' not
  found.com.ibm.ws.sib.msgstore.MessageStoreRuntimeException:
  Database, database_name, DB2 Authentication Error
  ```

  ```
  CWSIS1501E: The data source has produced an unexpected exception:
  java.sql.SQLException: Failed to start database 'database', see the
  next exception for details.
  ```

- If you find this exception, see 17.3.3, "The file store directories were created incorrectly" on page 452.

  ```
  File IO error: Permission Denied
  ```

- The following messages can indicate a configuration problem with the table schema, or can indicate a problem obtaining a lock during failover.

  ```
  […]CWSIS1535E: The messaging engine's unique id does not match that
  found in the data store.

  CWSIS1519E: Messaging engine TestCluster.001-bus1 cannot obtain the
  lock on its data store, which ensures it has exclusive access to the
  data.
  ```

  If you see these messages and do not believe you have a failover situation, or if you are not sure, see 17.3.4, "Table schema names are not unique" on page 453.

  If you see these messages and a messaging engine does not start after a failover, see 17.4.3, "Data store is locked" on page 458.

- If you find the following exception information, see 17.3.2, "Environment variables in the classpath are scoped incorrectly" on page 450.

  ```
  com.ibm.ws.sib.msgstore.MessageStoreRuntimeException:
  ClassNotFoundException class_name e.g a com.ibm.db2.jcc class
  ```

17.3 Clustering configuration problems

The following are some of the more common configuration errors that can prevent the messaging engine from starting.

17.3.1 JDBC provider or data source configuration error

Messages CWSIS1524E and CWSIS1501E usually indicate a problem that can be traced to a configuration error for the messaging data store JDBC provider or data source. Common mistakes are:

- The JDBC provider is configured at the wrong scope.
- The message store database does not exist.
- The authentication alias has not been specified correctly.
- The JNDI name is specified incorrectly.

The specific cause might be apparent from the exception messages. If so, you can go directly to one of these topics. If not, check each possibility.

The JDBC provider is configured at the wrong scope

The JDBC provider and data source defined for the messaging engine data store must be defined at the *Cell* or *Cluster* scope so they are visible to all cluster members. If defined at the node level, only the cluster members on that node will be able to use them.

This problem is indicated by the messages shown in Example 17-3.

Example 17-3 Messaging engine exception: Data source not found

```
[…] CWSIS0002E: The messaging engine encountered an exception while
starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: CWSIS1524E: Data
source, DefaultDataSource, not found.
```

You can resolve this problem by defining the JDBC provider for the messaging engine data store at the correct scope.

In an HA configuration, there will be one messaging engine for the cluster. In a WLM configuration, there will be multiple messaging engines for the cluster, each with its own distinct data store; therefore, you need to check the configuration for each JDBC provider and data source.

Using the administrative console:

1. Select **Resources** → **JMS** → **Data sources**.
2. Select **All scopes**.

   Identify the data sources for all messaging engines in the cluster and the scopes to which they are each defined.

   In an HA configuration, there is one messaging engine that can start on any cluster member. Ensure the data source and JDBC provider are defined at the Cluster or Cell scope so that each member of the cluster has access to that same definition.

   In a WLM configuration, each cluster member can have an active messaging engine. Ensure the data source for each messaging engine's message store is at a scope available to the cluster member and that it is unique.
3. If necessary, create a new JDBC provider and data source at the correct scope:
   a. Select the correct scope.
   b. Select **New**.
   c. Create a new JDBC provider (if necessary) and data source similar to the original resources, but with a new name and JNDI name.
4. Update the messaging engine with the new data source JNDI name:
   a. Select **Cluster** → ***cluster_name***.
   b. Select **Messaging Engine** → ***Messaging_engine***.
   c. Select **Message Store**.
   d. Update the data source JNDI name to refer to the newly defined data source.
   e. Save all changes and restart the cluster.

The message store database does not exist

If the database does not exist or if the data source has been configured with the incorrect database name, access to the database fails and the messaging engine cannot start.

This problem shows up as a statement in the SystemOut log as in Example 17-4 on page 447 suggesting the database cannot be found.

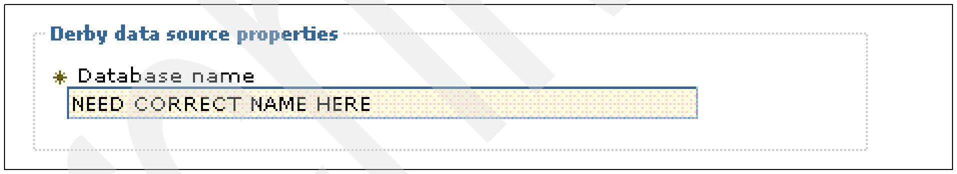
Example 17-4 Messaging engine exception: database not found

```
[…] 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.SQLException: Database 'SazFillin' not found.
```

Check the data source properties to make sure that the database name in the configuration matches the database name.

Using the administrative console:

1. Select **Resources** → **JDBC** → **Data sources.**
2. Select **All scopes**.
3. Identify the data sources for each messaging engine in the cluster. In an HA configuration, there is one messaging engine that can start on any cluster member. In a WLM configuration, there are multiple messaging engines, up to one per cluster member.
4. For each data source:
   a. Click the data source name to open the configuration page.
   b. Verify the database name is set to the correct database (Figure 17-1).


Derby data source properties

* Database name

NEED CORRECT NAME HERE

Figure 17-1 Database name setting in the data source

5. Save all changes and restart the cluster.

The authentication alias has not been specified correctly

This problem is specific to using a remote DB2 installation as the relational database for the messaging data store. A JAAS authentication alias associated with the data source is required for a connection to be made to the database.

If the JAAS authentication alias is not correct or is not set, access to the database fails. The symptom is a DB2 authentication error in the SystemOut log, similar to Example 17-5 on page 448.

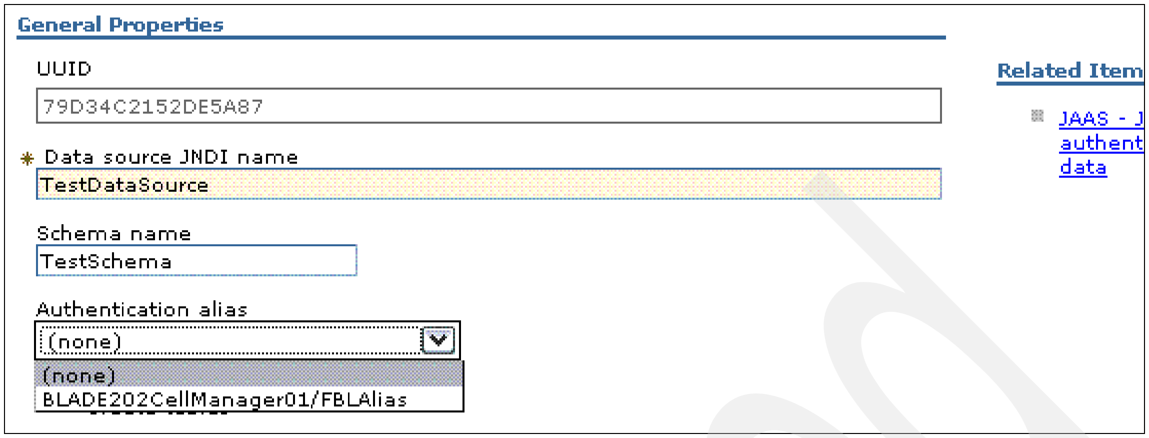
Example 17-5 Messaging engine exception: database authentication error

```
[…] CWSIS0002E: The messaging engine encountered an exception while
starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: Database, TestDB,
DB2 Authentication Error
```

The solution is to configure or correct the JAAS authentication alias.

Using the administrative console:

1. Select **Resources** → **JDBC** → **Data sources**.
2. Identify the data sources for each messaging engine in the cluster. In an HA configuration, there is one messaging engine that can start on any cluster member. In a WLM configuration, there are multiple messaging engines, up to one per cluster member.
3. For each data source:
   a. Click the data source name to open the configuration page.
   b. Click **JAAS-J2C authentication Data**.
   c. Either select the JAAS authentication alias of interest and check the input data is correct, or define a new JAAS authentication alias by selecting **New**.
4. If a new JAAS authentication alias is defined, you need to associate it with the messaging engines for the cluster.
   a. Select **Servers** → **Clusters** → ***cluster_name***.
   b. Select **Messaging engines** → ***messaging_engine.***
   c. Select **Message store.**
   d. Use the drop-down to select the authentication alias, as shown in Figure 17-2 on page 449.


General Properties

UUID

79D34C2152DE5A87

* Data source JNDI name

TestDataSource

Schema name

TestSchema

Authentication alias

(none)

(none)

BLADE202CellManager01/FBLAlias

Figure 17-2 Select the authentication alias

5. Save all changes and restart the cluster.

The JNDI name is specified incorrectly

The JNDI name specified in the data source must match that specified in the messaging engine configuration. If it does not match, the exception shown in Example 17-6 will occur.

Example 17-6 Messaging engine exception: Failed to start database

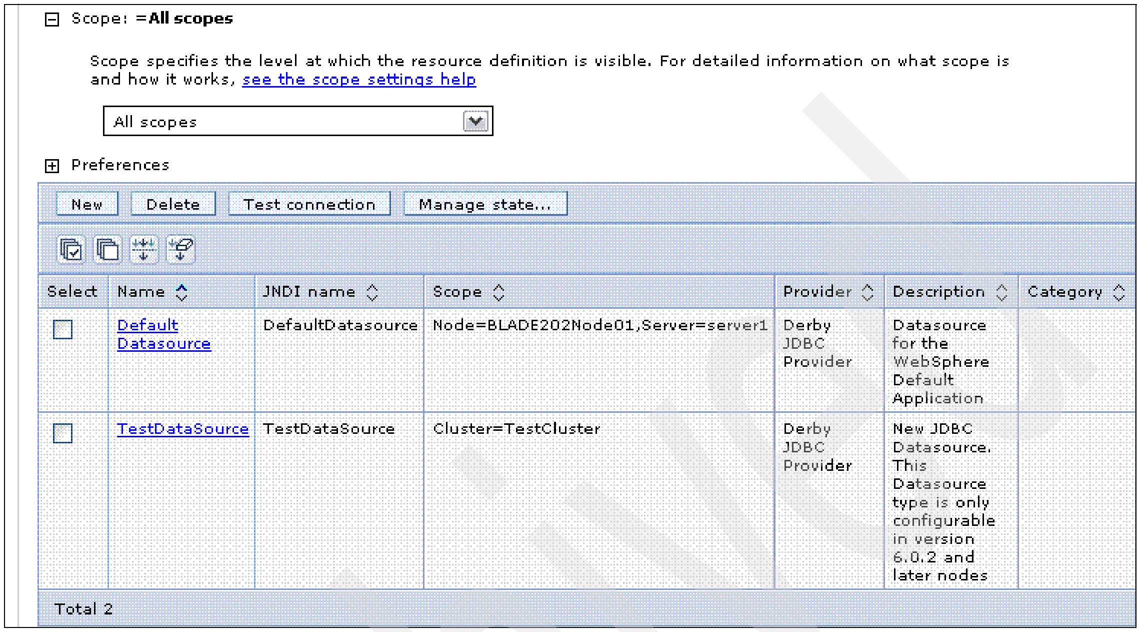
```
[…]CWSIS0002E: The messaging engine encountered an exception while
starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException: CWSIS1524E: Data
source, DefaultDataSource, not found.

[…]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.SQLException: Failed to start database
'c:/was61/profiles\AppSrv01/installedApps/BLADE202Cell01/DefaultApplica
tion.ear/DefaultDB', see the next exception for details.
```

To resolve this problem, check and correct (if necessary) the data source properties for each messaging engine.

Using the administrative console:

1. Select **Resources** → **JDBC** → **Data sources**
2. Set the scope to **All Scopes.**

You see a list of all the data sources defined at various scopes, similar to the list shown in Figure 17-3.


⊟ Scope: =**All scopes**

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help

All scopes

⊞ Preferences

Select	Name	JNDI name	Scope	Provider	Description	Category
☐	Default Datasource	DefaultDatasource	Node=BLADE202Node01,Server=server1	Derby JDBC Provider	Datasource for the WebSphere Default Application	
☐	TestDataSource	TestDataSource	Cluster=TestCluster	Derby JDBC Provider	New JDBC Datasource. This Datasource type is only configurable in version 6.0.2 and later nodes	

Total 2

Figure 17-3 List of data sources

3. Make a note of the data source name and JNDI name for each cluster member messaging engine. If there is not a data source defined for each cluster member, you need to define them.
4. Next, use the information you collected to make sure the messaging engine has been configured with the correct JNDI name. Select **Servers** → **Clusters** → ***cluster_name***.
5. Select **Messaging engines** → ***messaging_engine***.
6. Select **Message store**.
7. Ensure the data source JNDI name is correct for that messaging engine.
8. Repeat this procedure for each cluster member that has its own database.
9. Save all changes, and restart the cluster.

17.3.2 Environment variables in the classpath are scoped incorrectly

Problems can occur when the JDBC provider for a messaging engine data store uses WebSphere variables in the classpath definition, and the variable does not exist or it is defined at the wrong scope.

This problem occurs when drivers for the JDBC provider are stored in different locations on the systems hosting the cluster members. If the variable is only defined at the Cell or Cluster scope, it will be incorrect for at least one of the cluster members.

This problem shows up as a statement in the SystemOut log as shown in Example 17-7.

Example 17-7 Messaging engine exception: ClassNotFoundException

```
[…] CWSIS0002E: The messaging engine encountered an exception while
starting. Exception:
com.ibm.ws.sib.msgstore.MessageStoreRuntimeException:
ClassNotFoundException <name of class>  (for example, com.ibm.db2.jcc
class)
```

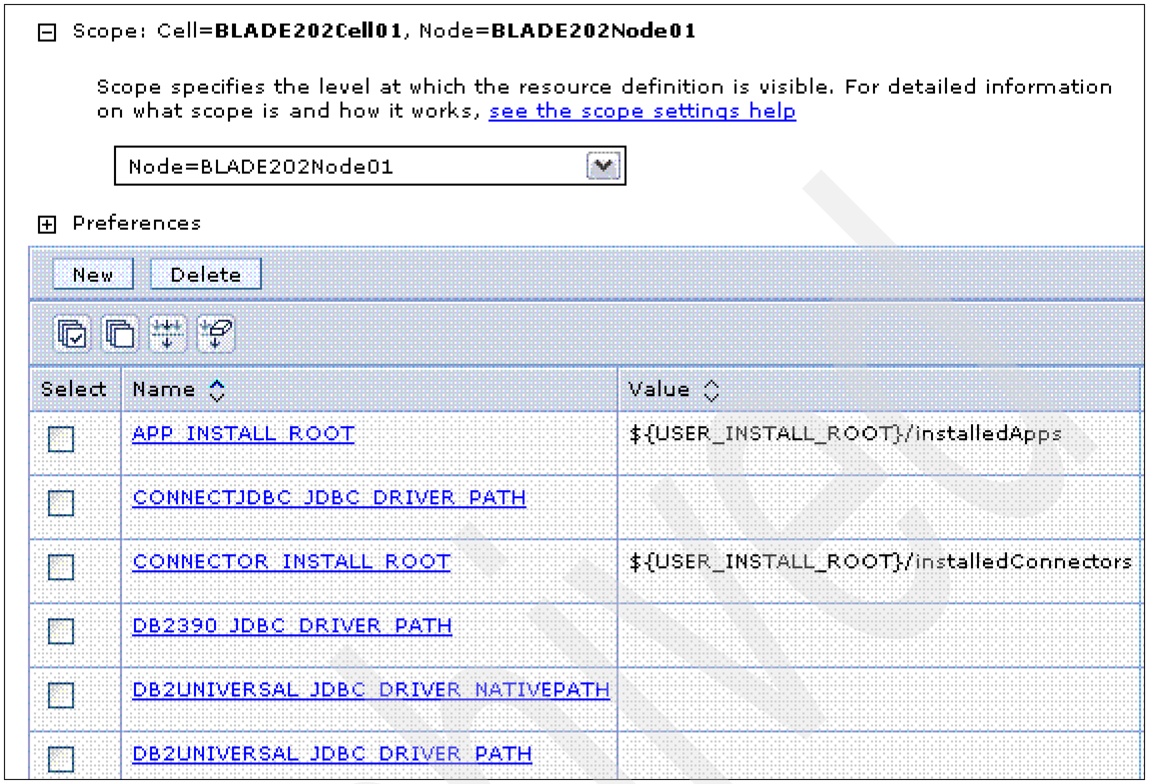
To resolve this problem, define the WebSphere variable correctly on each cluster member.

For example, a configuration using a DB2 universal database as a data store needs the location of the DB2 JDBC drivers defined for each cluster member. Therefore, you need to define the DB2UNIVERSAL_JDBC_DRIVER_PATH variable.

To define this variable using the administrative console:

1. Select **Environment** → **WebSphere Variables**.
2. Select the correct scope from the drop-down.

   You see a list of variables as shown in Figure 17-4 on page 452.


Scope: Cell=**BLADE202Cell01**, Node=**BLADE202Node01**

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help

Node=BLADE202Node01

Preferences

Select	Name	Value
☐	APP_INSTALL_ROOT	${USER_INSTALL_ROOT}/installedApps
☐	CONNECTJDBC_JDBC_DRIVER_PATH	
☐	CONNECTOR_INSTALL_ROOT	${USER_INSTALL_ROOT}/installedConnectors
☐	DB2390_JDBC_DRIVER_PATH	
☐	DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH	
☐	DB2UNIVERSAL_JDBC_DRIVER_PATH	

Figure 17-4 WebSphere environment variables for DB2

3. Verify and update the variables as needed to ensure they have the correct value. For example, DB2UNIVERSAL_JDBC_DRIVER_PATH must have the value of the location of the DB2 JDBC drivers on the cluster member of interest.

   If the variable does not exist at the scope, create it by clicking **New** and completing the Name and Value fields.

4. Save all changes and restart the cluster.

17.3.3 The file store directories were created incorrectly

If a file store is used for messaging engine persistence, problems can occur if the log and storage directories are created by the user rather than by the messaging engine. A messaging engine creates the directories it needs as it starts up.

This problem shows up as a statement in the SystemOut log with the following message text:

```
File IO error: Permission Denied
```

To solve this problem, remove the directories, which you can find in the WebSphere directory structure:

*app_server_root/*Profiles/*cluster_profile*/Filestores/com.ibm.ws.sib/

Restart the cluster and the messaging engines will create the directories it needs on start up.

17.3.4 Table schema names are not unique

If a single database is used to house data store tables for multiple messaging engine tables, each messaging engine must be assigned a unique schema name to avoid two messaging engines attempting to obtain a lock on the same table set.

This problem shows up in the SystemOut log as shown in Example 17-8.

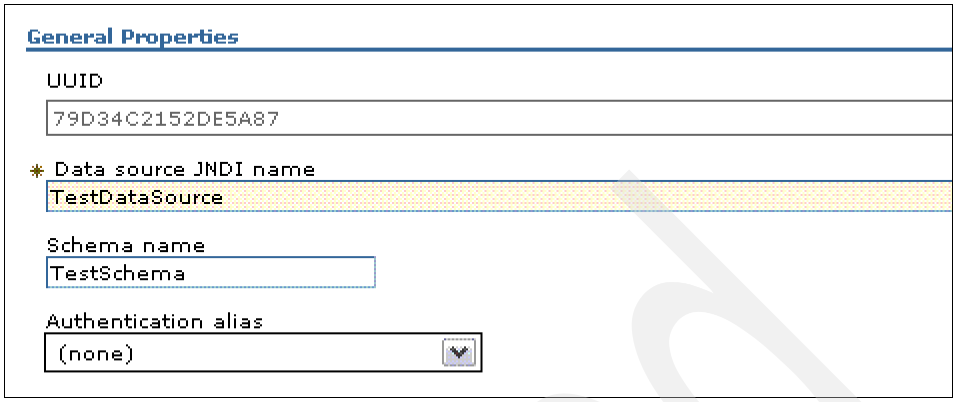
Example 17-8 Messaging engine exception: cannot obtain lock on data store

```
[…]CWSIS1535E: The messaging engine's unique id does not match that
found in the data store.
CWSIS1519E: Messaging engine TestCluster.001-bus1 cannot obtain the
lock on its data store, which ensures it has exclusive access to the
data.
```

To correct this problem, create unique schema names for each cluster member.

Using the administrative console:

1. Select **Service Integration** → **Buses**.
2. Click on the bus name.
3. Click **Messaging Engines**.
4. Click on the messaging engine name.
5. Click **Message Store**.
6. Make sure the Schema name box is filled with a unique schema name as shown in Figure 17-5 on page 454.


General Properties

UUID

79D34C2152DE5A87

* Data source JNDI name

TestDataSource

Schema name

TestSchema

Authentication alias

(none)

Figure 17-5 Messaging engine data store schema name

7. Repeat this procedure for each of the messaging engines, making sure all schema names are unique.
8. Save the configuration changes and restart the cluster.

17.4 Cluster runtime problems

This section discusses problems specific to clustering of messaging engines where the messaging engines start correctly but there is a problem with message delivery.

17.4.1 Core group policy is incorrect or no preferred server defined

To exert some control over the location of the messaging engine, you can define a preferred server in the core group policy. The messaging engine will start on the preferred server if the cluster member is active.

If a core group policy is defined incorrectly with invalid or non-matching criteria, the WLM manager randomly assigns the messaging engines to cluster members, resulting in the messaging engines being started on random cluster members.

Defining a preferred server

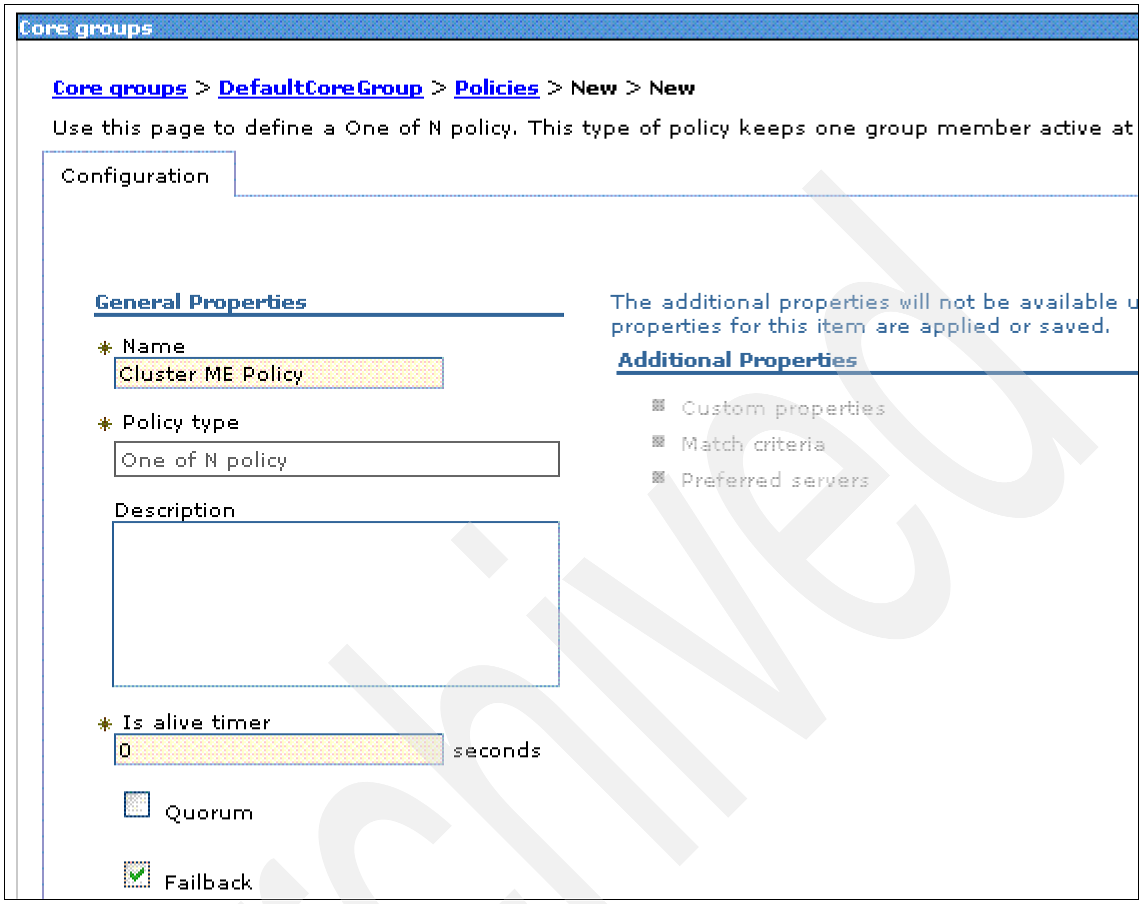
First, define a core group policy that can be associated with a messaging engine. The default messaging provider includes a DefaultCoreGroup that can be extended for cluster configurations.

To create a core group policy using the administrative console:

1. Select **Servers** → **Core Groups** → **Core group settings.**
2. Click on the core group name.
3. In the Configuration tab, select **Policies**.
4. Click **New** to define a new policy (Figure 17-6 on page 456):
   a. Select **One of N** as the policy type.

      The One of N policy keeps one member of the high availability group active at all times. It is used by groups that desire singleton failover. In an HA configuration, there is only one messaging engine.

   b. Enter a name for the policy.
   c. If you want failback to be active (that is, the messaging engine fails back when the preferred server comes online after a failover has taken place), select the failback box, as shown in Figure 17-6 on page 456.


Core groups

Core groups > DefaultCoreGroup > Policies > New > New

Use this page to define a One of N policy. This type of policy keeps one group member active at

Configuration

General Properties

* Name
Cluster ME Policy

* Policy type
One of N policy

Description

* Is alive timer
0 seconds

☐ Quorum

☑ Failback

The additional properties will not be available u
properties for this item are applied or saved.

Additional Properties

- Custom properties
- Match criteria
- Preferred servers

Figure 17-6 Core group policy for high availability and failback

Next, you define the match criteria that associates a messaging engine with the policy.

Using the administrative console:

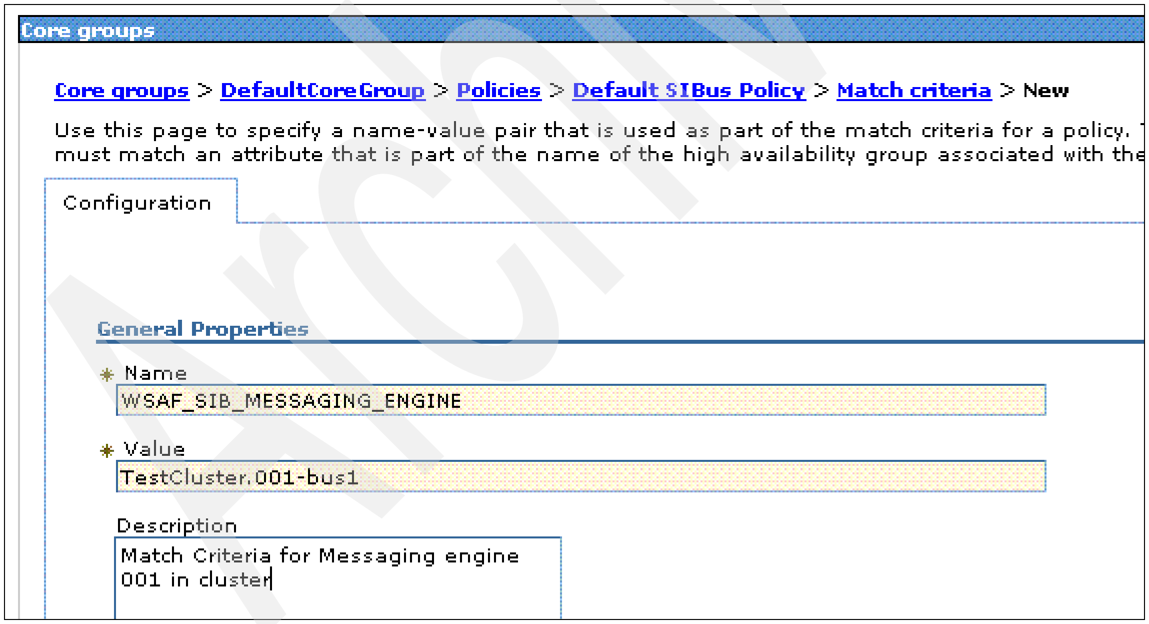
1. Select **Servers** → **Core groups** → **Core group settings**.
2. Click on the core group.
3. In the configuration tab, select **Policies.**
4. Click on the policy name.
5. Select **Match Criteria**.
6. Click **New**.

7. Enter the name/value pair information. The Name and Value fields together specify a messaging engine or group of messaging engines, it will be linked to this policy. Name/value pairs that you can use shown in Table 17-1.

Table 17-1 Match criteria name/value pairs

Name	Value	Messaging engines the policy will match
type	WSAF_SIB	Any messaging engine
WSAF_SIB_MESSAGING _ENGINE	The name of your messaging engine.	A particular messaging engine
WSAF_SIB_BUS	The name of your bus	All messaging engines in a particular bus
IBM_hc	The name of your cluster	All messaging engines in a particular cluster

In this instance, you need to use the WSAF_SIB_MESSAGING_ENGINE name/value pair, as shown in Figure 17-7.


Core groups

Core groups > DefaultCoreGroup > Policies > Default SIBus Policy > Match criteria > New

Use this page to specify a name-value pair that is used as part of the match criteria for a policy. T must match an attribute that is part of the name of the high availability group associated with the

Configuration

General Properties

* Name
WSAF_SIB_MESSAGING_ENGINE

* Value
TestCluster.001-bus1

Description
Match Criteria for Messaging engine 001 in cluster

Figure 17-7 Name / value pair

Finally, specify the preferred server for the policy:

1. Select **Servers** → **Core groups** → **Core group settings**.

2. Click on the core group name.
3. In the configuration tab select **Policies.**
4. Click on the policy name.
5. Click **Preferred servers**.
6. Select the cluster member on which the messaging engine should start , and use **Add** to move the cluster member to the preferred server list (on the right).

   Multiple servers can be in the list but the order is important. There is an implicit stronger preference for a server that appears earlier in the list.
7. Save the changes and restart the cluster.

17.4.2 A permanent reply queue is in use

New in WebSphere V6.1 is the ability to create and assign a permanent reply queue for JMS response messages. This functionality, while supported in a single server environment, is not supported in a clustered environment because the response message currently cannot be guaranteed to use the same partition of the destination as the request message.

17.4.3 Data store is locked

When a messaging engine tries to fail over and cannot complete the process, a problem can occur with the data store lock. If the messaging engine is taking over from another that was running on a different cluster member, the database might not have released the database locks that comprise the datastore lock.

This problem shows up as a statement in the SystemOut log suggesting that a lock cannot be obtained on the data store as in Example 17-9.

Example 17-9 Failover fails due to data store lock

```
[…]CWSIS1535E: The messaging engine's unique id does not match that
found in the data store.

CWSIS1519E: Messaging engine TestCluster.001-bus1 cannot obtain the
lock on its data store, which ensures it has exclusive access to the
data.
```

To resolve this problem, use the administration tools for the chosen relational database management system to examine the locks on the SIBOWNER table. If the database is still holding locks after the failure of a cluster member, release the locks and attempt to restart the messaging engine on the appropriate cluster member, using the administrative console.

If a foreign bus is defined it is also affected by the failover process. After the messaging engine is restarted, the foreign bus should restart. For more information about foreign bus problem determination, see Chapter 24, “Foreign bus problem determination” on page 529.

17.4.4 Orphaned messages after failover

An orphaned message is one that exists on a destination but no application consumes it. You would normally determine that you have orphaned messages by monitoring the number of messages sent as opposed to the number of (expected) messages received. You might also notice from the administrative console that the message depth of a queue is not zero when you would expect it to be.

Reasons that messages can be orphaned after a failover include:

- If a messaging engine fails over from one active cluster member to another active cluster member, messages keep flowing; however, if failover cannot take place, persistent messages can become orphaned.

  To receive the orphaned messages, the messaging engine needs to be started on one of the cluster members. At this point, a client can connect and consume the messages. This state is applicable to all destination points.

  To start a messaging engine, see 10.2.3, “Determine your messaging engine status” on page 328.
- If a foreign bus is defined, there might be orphaned messages due to one end of the link not failing over. For more information about foreign bus problem determination, see Chapter 24, “Foreign bus problem determination” on page 529.
- Messages can also be orphaned after a successful failover if the client is not configured correctly. If cluster members are spread across multiple machines in an HA configuration, any clients need to be able to connect to all of the cluster members.

  To solve this problem, create an endpoint list.

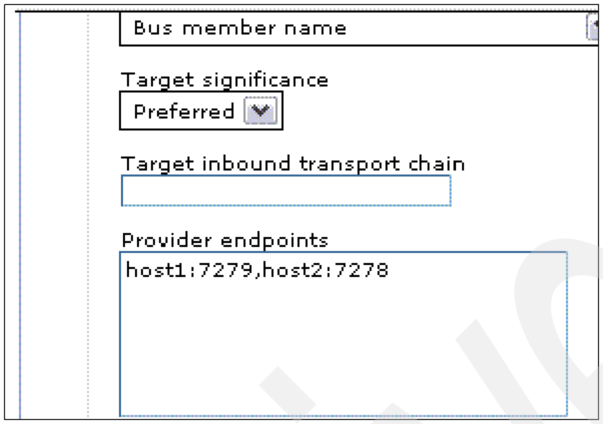
Creating an endpoint list

To create an endpoint list using the administrative console:

1. Select **Resources** → **JMS**.
2. Select **Connection Factories**.

3. Select the ***connection_factory_name*** that the client is trying to use to connect to the cluster.

   You can enter a comma separated list of provider endpoints into the provider endpoints box as shown in Figure 17-8.



Bus member name

Target significance: Preferred

Target inbound transport chain

Provider endpoints: host1:7279,host2:7278

Figure 17-8 Provider endpoints

   The list takes the form *host:port:chain*,*host:port:chain*. Enter all of the cluster members that are in a highly available group:

   – *host* is the box where a particular cluster member is located.

   – *port* is the SIB_ENDPOINT_ADDRESS or SIB_ENDPOINT_SECURE_ADDRESS for the cluster member (see **Servers** → ***server_name*** → **Ports**).

   – *chain* is the target transport chain and is optional in this list.

4. Save your changes and restart cluster.

17.4.5 Core group policy is not correct or no preferred server defined

If a core group policy is incorrectly defined with invalid or non matching criteria, the WLM manager randomly assigns the messaging engines to cluster members. In a WLM configuration, this can result in multiple messaging engines being started on a single cluster member, while leaving other cluster members with no messaging engine.

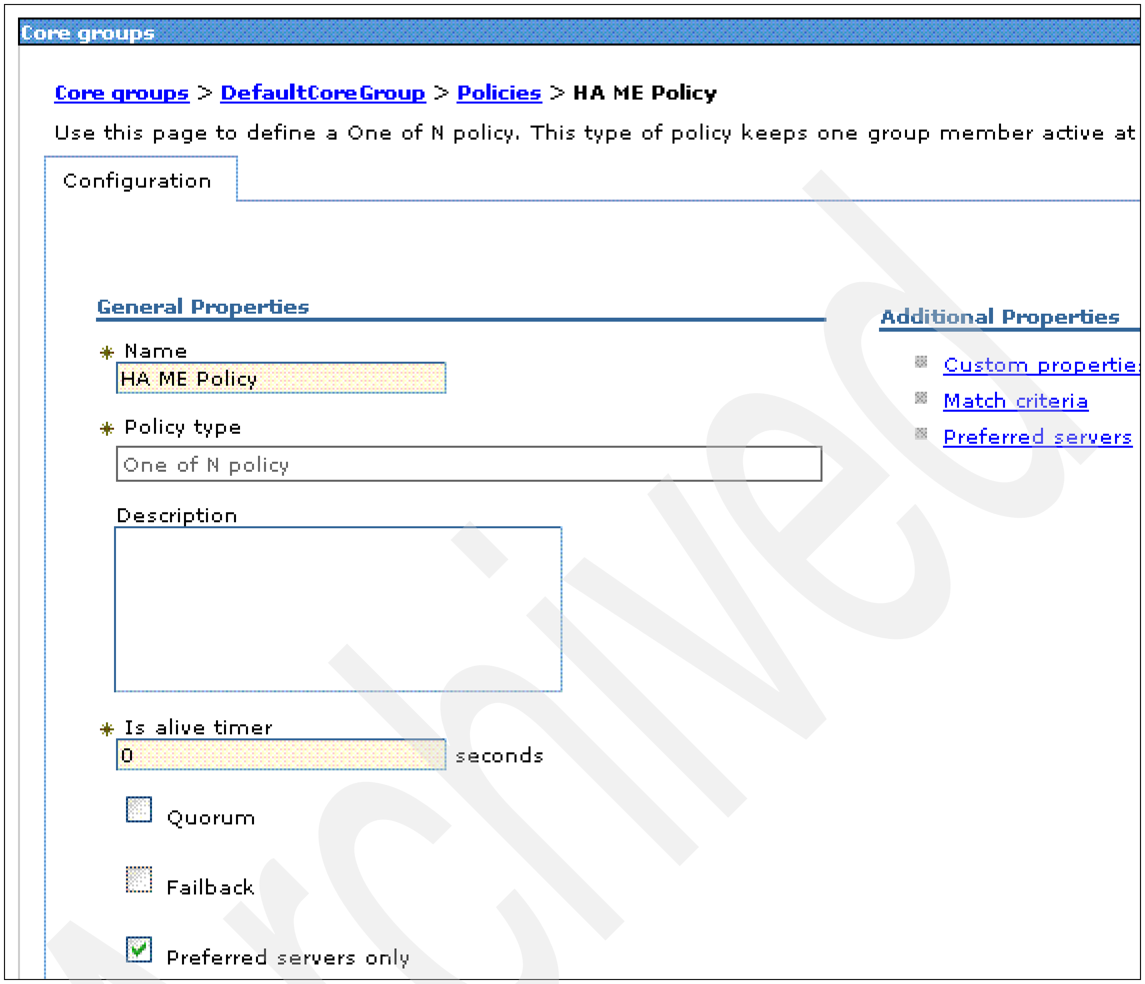
The solution is to pin the messaging engines to cluster members using core group policies. The messaging engines will then start on the defined cluster members if the cluster members are active.

Some WLM clusters have highly available messaging engines in the cluster. The HA messaging engines have a preferred server defined instead of being pinned to a cluster member; other messaging engines that are not HA are pinned.

To define a preferred server, see 17.4.1, “Core group policy is incorrect or no preferred server defined” on page 454. The process to pin a messaging engine to a cluster member in a WLM configuration is similar; however, make sure to select the **Use preferred servers only** box when you define a `One of N policy`, as shown in Figure 17-9 on page 462.

This definition pins the messaging engine to the preferred server so that it cannot fail over. If the cluster member goes offline, the messaging engine is not available until the cluster member comes back online. You can also use a static policy to pin a messaging engine.

The WLM cluster has multiple messaging engines, so you need to perform the process described above for each messaging engine.


Core groups

Core groups > DefaultCoreGroup > Policies > HA ME Policy

Use this page to define a One of N policy. This type of policy keeps one group member active at

Configuration

General Properties

* Name
HA ME Policy

* Policy type
One of N policy

Description

* Is alive timer
0 seconds

☐ Quorum

☐ Failback

☑ Preferred servers only

Additional Properties

- Custom propertie
- Match criteria
- Preferred servers

Figure 17-9 Select Preferred servers only

17.4.6 Orphan messages on a partitioned destination

In a WLM configuration, with multiple messaging engines in a cluster, a queue destination becomes partitioned. The number of partitions depends upon the number of messaging engines because every messaging engine in the cluster has a queue point. If the destination is a topic space, it has a publication point localized to every messaging engine in the cluster. For more information about destination points, see 10.2.4, “Finding queued messages” on page 329.

Partitions cannot communicate with other partitions. A client must be connected to each cluster member to collect all messages from all partitions.

If an active cluster member goes offline, persistent messages being routed through the cluster member can remain on the destination point until the cluster member and messaging engine becomes active again. After the cluster member becomes active again, the client can reconnect and consume the messages.

17.4.7 Cluster weights incorrectly specified

If the weights are incorrectly specified in a WLM cluster configuration some messages might not arrive on a specified partition of a partitioned destination. If a weight of `0` is specified for a particular cluster member, messages will not flow to the destination point on this cluster member. This configuration variable is not an attribute of messaging; it is an attribute of the server.

For information about specifying cluster weights, see:

- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- *Clusters and workload management* at:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/crun_srvgrp.html`

18

Message-driven beans problem determination

This chapter discusses common problems you might encounter when using a message-driven bean (MDB) configured with WebSphere Application Server default messaging.

If you are using MDBs to consume messages from a WebSphere MQ destination, see Chapter 19, “WebSphere MQ and MDBs” on page 477.

18.1 Introduction to message-driven beans

Message-driven beans (MDBs) are EJBs that act as message consumers. An MDB always runs inside an application server, and it connects to a messaging engine to consume messages. MDB applications are invoked by WebSphere when messages arrive on the queue the MDB has been configured to consume from.

When an MDB application starts:

1. The activation specification is read and validated. If the activation specification fails validation the MDB will not start.
2. The MDB attempts to connect to a messaging engine in the bus that has been specified on the activation specification. If there are no messaging engines available then one of the following occurs:
   - If the MDB is running on a server that is a bus member which the activation specification references, then it will wait for the messaging engine on that same server to start and connect to it.
   - If the MDB is running on a server that is not a bus member of the specified bus, then it will retry every 30 seconds to connect to a messaging engine that might be running on a remote server.
3. The MDB creates an asynchronous consumer on the configured destination.
4. The consumer informs the MDB application whenever a message is available for consumption by calling its `onMessage` method.

For more information

For more information about activation specification settings, see:

- *JMS activation specification settings* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/sibjmsresources/SIBJMSActivationSpec_DetailForm.html

- If your MDB is attempting to consume messages from a remote WebSphere Application Server Cell, some additional configuration steps are necessary.

  See *Configuring the core group bridge service* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_ha_coregroupbridge.html

- Message ordering considerations

  You can configure MDBs to process multiple messages in parallel or to have batches of messages sent for performance benefits. However, if message ordering is important then you must have messages processed one at a time, by a single MDB instance. To achieve this behavior, you need to carefully select the activation specification properties.

See *JMS activation specification settings* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc\sibjmsresources\SIBJMSActivationSpec_DetailForm.html

- MDBs and cluster considerations

  For more information, see Section 9.5.2, *MDBs and clusters* in *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304

18.1.1 Symptoms of a messaging-driven bean problem

Symptoms addressed in this chapter include:

- A message-driven bean application fails to start.
- A message-driven bean fails to connect to a messaging engine.
- A message-driven bean fails to consume messages.
- A message-driven bean fails during processing.
- Messages are lost.

18.2 Verify integrity

If you are having problems related to a messaging-driven bean application, do the following steps before proceeding with problem determination:

- Verify the MDB application is started.
- Verify that messages can be sent to the target destination.
- Verify the MDB is consuming messages.

18.2.1 Verify the MDB application is started

To diagnose the problem, first determine the state of the deployed MDB application. The simplest method to check the state is using the administrative console:

1. Select **Applications** → **Enterprise Applications**.

You see a list of deployed applications along with their status. If the MDB application has not started, a red cross displays in the corresponding status column. If the application is started, then a green arrow displays.

2. If the application is not started, select the box to the left of the application and click **Start**.

If the MDB application fails to start, see 18.3, “Analyze the SystemOut log for the MDB server” on page 469.

18.2.2 Verify that messages can be sent to the target destination

Verify that messages can be produced for the target destination. For example, create a test application to produce messages that are sent to the destination.

18.2.3 Verify the MDB is consuming messages

If the MDB fails to handle a message, the messaging engine attempts to redeliver the message based on the destination properties of the destination. To display these properties, use the administrative console to:

1. Select **Service Integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations** to display a list of queue and topic destinations.
4. Select the appropriate queue destination from the displayed list.

If the message cannot be processed, delivery is reattempted until the maximum threshold for failed deliveries is met. At this point, the messaging engine attempts to deliver the message to the exception destination. By default, this address uses a destination that is generic for that messaging engine. The naming format of the default exception destination is:

```
SYSTEM.Exception.Destination.Messaging_Engine_Name
```

Depending on your configuration, a unique exception destination might have been defined for the queue in question.

By default, no application server log messages are produced when an MDB processes a received message. With this in mind, it is advisable that you add some form of logging to an application so that you can recognize that messages are being processed successfully.

18.3 Analyze the SystemOut log for the MDB server

Examine the application server log files for any exceptions. The most common cause of problems is a configuration problem with the activation specification the MDB is configured to use.

If the SystemOut log includes:

- J2CA0164E, see 18.4.1, “Activation specification destination name incorrect” on page 470.
- J2CA0052E, see 18.4.2, “Unable to locate the activation specification” on page 471.
- J2CA0137E, collect any error messages following (CWSJR*xxxx*E) and see 18.4.3, “Invalid activation specification” on page 471.
- The following messages, see 18.4.4, “Listener port used for default messaging resources” on page 472:

  ```
  Unable to start MDB Listener QueueReceiver, JMSDestination
  Destination_JNDI_Name :
  com.ibm.ejs.jms.listener.MDBInvalidConfigException: Cannot deploy an
  MDB against a listener port specifying a default messaging JMS
  resource.
  ```
- CWSIV0775W, see 18.4.5, “MDB unable to connect to a messaging engine” on page 473:
- CNTR0020E, this message provides the primary source of problem determination information for MDB processing problems.

  See 18.4.6, “MDB does not handle exception appropriately” on page 473.

If the MDB is not consuming messages but you find no exception information in the SystemOut log, see 18.4.7, “MDB started but inactive: No errors in the logs” on page 474.

18.4 Causes and solutions

Generally, MDB problem determination can be categorized into the following areas:

- A message-driven bean application fails to start.

  This failure is generally caused by a configuration problem. For possible causes, see:

  – 18.4.1, “Activation specification destination name incorrect” on page 470.

  - 18.4.2, “Unable to locate the activation specification” on page 471.
  - 18.4.3, “Invalid activation specification” on page 471.
- A message-driven bean fails to consume messages. See:
  - 18.4.4, “Listener port used for default messaging resources” on page 472.
  - “Activation specification used for WebSphere MQ resources” on page 471.
  - 18.4.7, “MDB started but inactive: No errors in the logs” on page 474.
- A message-driven bean fails to connect to a messaging engine.

  See 18.4.5, “MDB unable to connect to a messaging engine” on page 473.
- A message-driven bean fails during processing.

  See 18.4.6, “MDB does not handle exception appropriately” on page 473.

18.4.1 Activation specification destination name incorrect

The symptom of this problem is this message in Example 18-1.

Example 18-1 Activation specification error J2CA0164E

```
ActivationSpe E   J2CA0164E: The lookup of the Destination with JNDI
Name JNDI_Name  failed. The Destination is required by the Activation
Specification. The lookup failed due to exception
javax.naming.NameNotFoundException: Context: Scope_Path, name: JNDI
Name: First component in name JNDI_Name not found. [Root exception is
org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0]
```

Check that the destination name is correct and that the activation specification and destination JMS resources have been bound into JNDI.

To find JMS resources, look in **Resources** → **JMS** → ***destination_type***. Select the destination to view the JNDI name, which must match the destination JNDI name specified in the activation specification for the destination. The destination might also be specified at deploy time or by the application developer in the configuration file for MDB (inside the EAR file).

To find activation specifications, look in **Resources** → **JMS** → **Activation specifications**.

18.4.2 Unable to locate the activation specification

The symptom of this problem is this message in Example 18-2.

Example 18-2 Activation specification error J2CA0052E

```
ActivationSpe E   J2CA0052E: The lookup of the Activation Specification
with JNDI Name  JNDI_Name  failed due to the following exception:
javax.naming.NameNotFoundException: Context: Scope_Path, name:
JNDI_Name : First component in name JNDI_Name   not found. [Root
exception is org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0]
```

Check that the activation specification that the MDB has been configured to use exists. Either the MDB was configured to use an incorrect activation specification or the activation specification has not been created.

18.4.3 Invalid activation specification

Message J2CA0137E indicates an invalid property has been specified for the activation specification. This message contains nested messages that provide more information about the failure. See Example 18-3.

Example 18-3 Invalid activation specification property

```
ActivationSpe E   J2CA0137E: The ActivationSpec validate() method
failed with an InvalidPropertyException.  The ActivationSpec is
ActivationSpecification_JNDI_Name, which belongs to the installed
ResourceAdapter RA_Name and is associated with the MDB Application
MDB_Application_Name.  See the following list of failed properties
along with their values: Property_Name.   The exception
is:javax.resource.spi.InvalidPropertyException: CWSJR1181E: The JMS
activation specification has invalid values - the reason(s) for failing
to validate the JMS activation specification are: Nested_Exception(s)
```

This message occurs when a problem with the configuration has been detected. See the nested exception for more information, and update the activation specification property information.

Activation specification used for WebSphere MQ resources

One specific example of this type of error can occur when an MDB is configured to consume from a WebSphere MQ destination using an activation specification.

Example 18-4 on page 472 illustrates the error that can occur.

Example 18-4 Invalid use of activation specification

```
ActivationSpe E   J2CA0137E: The ActivationSpec validate() method
failed with an InvalidPropertyException.  The ActivationSpec is
Activation_Specification_JNDI_Name, which belongs to the installed
ResourceAdapter RA_Name and is associated with the MDB Application
MDB_Application_Name.  See the following list of failed properties
along with their values: destination Destination_Name. The exception
is: javax.resource.spi.InvalidPropertyException: CWSJR1181E: The JMS
activation specification has invalid values - the reason(s) for failing
to validate the JMS activation specification are: [CWSJR1192E: JMS
activation specs using a destination type of queue must have a
destination of type [com.ibm.websphere.sib.api.jms.JmsQueue] but the
destination passed was of type [com.ibm.mq.jms.MQQueue]]
```

Ensure that the MDB is configured with an activation specification and is pointing to a destination of the correct type. In this example, the destination reference is wrong. An activation specification is being used so the destination should refer to a service integration bus destination.

18.4.4 Listener port used for default messaging resources

A listener port has been configured for use with the default messaging resources. This problem is characterized by these messages present in the application server log files in Example 18-5.

Example 18-5 Unable to start MDB listener

```
Unable to start MDB Listener QueueReceiver, JMSDestination
Destination_JNDI_Name :
com.ibm.ejs.jms.listener.MDBInvalidConfigException: Cannot deploy an
MDB against a listener port specifying a default messaging JMS
resource.
```

MDBs that use default messaging must be configured to use an activation specification and not a listener port. Likewise, MDBs that are configured to consume messages from a WebSphere MQ messaging destination must be configured to use a listener port and not an activation specification.

18.4.5 MDB unable to connect to a messaging engine

The MDB is unable to create a connection to a messaging engine in the specified bus as in Example 18-6.

Example 18-6 MDB unable to connect to a messaging engine

```
SibMessage    W   [:] CWSIV0775W: The creation of a connection for
destination Destination_Name on bus Bus_Name for endpoint activation
Endpoint_Activation_Details failed with exception
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0086E: Bus
Bus_Name not found
```

If this problem occurs:

- Ensure the bus name in the activation specification is correct and is pointing to the correct bus for your configuration. You can find activation specifications under **Resources** → **JMS** → **Activation specifications**.
- Ensure the servers hosting the messaging engines in the specified bus are started.
- Ensure that the target messaging engines on the remote servers are started.

You might find additional diagnostic information in nested exceptions.

18.4.6 MDB does not handle exception appropriately

When an MDB is given a message to consume, the MDB takes on the responsibility of providing logging until the MDB's `onMessage` method completes. If an exception is thrown by the MDB, the application server catches the exception and traces it to the log files.

The message shown in Example 18-7 indicates an exception occurred in the MDB but was not handled.

Example 18-7 Exception occurs in an MDB

```
[...] 00000066 ExceptionUtil E CNTR0020E: EJB threw an unexpected
(non-declared) exception during invocation of method "onMessage" on
bean
"BeanId(PackageReceivedEAR#PackageReceived.jar#PackageReceived, null)".
Exception data: java.lang.RuntimeException: MDB error at
receiver.PackageReceivedBean.onMessage(Unknown Source)
```

```
at
com.ibm.ejs.container.MessageEndpointHandler.invokeMdbMethod(MessageEnd
pointHandler.java:990)
```

It is the responsibility of the MDB application to handle exceptions.

The solution in this case is to modify the MDB application to handle exception cases.

Message not restored to queue after MDB failure

The MDB transaction-type set in the EJB module deployment descriptor can be container-managed or bean-managed. A bean-managed transaction is responsible for committing or rolling back the transaction. It does this with the `UserTransaction` interface.

If the MDB is bean managed and it does not correctly cope with transactions and exceptions, messages be “lost.”

For more information, see Chapter 8 of *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304.

18.4.7 MDB started but inactive: No errors in the logs

If a messaging engine for the desired bus is defined on the local server, then the MDB only attempts to connect to the local messaging engine. If the local messaging engine is not currently running, the MDB will silently wait for it to start.

If an MDB is deployed to a cluster, the MDB application will start on all servers within the cluster, but only the MDB instance on the server where the messaging engine is active will consume messages. During failover, you might experience a period of time in which the MDB is waiting for a local messaging engine to start. Several defects in this area were fixed in fix pack 6.1.0.9 (and 6.0.2.23) for WebSphere Application Server.

18.5 Validate the solution

If you have made changes to the WebSphere Application Server JMS or JNDI resources, you must restart the application server processes for changes to take effect.

Verify the MDB application is started and that the MDB application is processing messages correctly.

19

WebSphere MQ and MDBs

This topic covers some of the problems that you might encounter with a message-driven bean configured to consume messages from a WebSphere MQ destination using a listener port.

19.1 Symptoms of problems with MDBs and listener ports

Symptoms of a problem with MDBs that consume messages from a WebSphere MQ destination using a listener port include:

- The MDB application does not start.
- The MDB fails to connect to the WebSphere MQ queue.
- The MDB application starts, but is not driven by messages on the WebSphere MQ destination.
- The SystemOut log contains `com.ibm.mqservices.MQInternalException` messages.
- The SystemOut log contains `javax.naming.NameNotFoundException` for the destination.
- The SystemOut log contains `javax.jms.InvalidDestinationException` for the destination.
- The SystemOut log contains `com.ibm.ejs.jms.listener.MDBInvalidConfigException` for the destination.

19.2 Verify system integrity

To ensure the environment is properly set up, work with the WebSphere MQ administrator:

- Ensure the target WebSphere MQ queue manager is available.
- Ensure messages can be produced for the target destination; for example, use WebSphere MQ Explorer.

19.3 Analyze the application server log

Examine the application server log for the MDB application; look for error messages prefixed with `WMSG`. The nested exception contains an error message indicating the reason for the failure to start the listener port.

If the root cause is a WebSphere MQ problem, a WebSphere MQ reason code is included in the linked exception information. For example:

```
:Caused by: com.ibm.mqservices.MQInternalException: MQJE001: An
MQException occurred: Completion Code 2, Reason 2059
MQJE011: Socket connection attempt refused
```

Tip: The **mqrc** ***reason_code*** command on a WebSphere MQ system returns a brief description of the given reason code.

These exceptions might occur:

- `JMSDestination` *`jms/theQ`*`: javax.naming.NameNotFoundException`

  The JMS destination provided to the listener port was not found in JNDI. Ensure that the destination JNDI name is specified correctly.

- `JMSDestination` *`jms/theQ`*`: javax.jms.InvalidDestinationException`

  A JMS queue that is not of type WebSphere MQ messaging provider has been specified on the listener port.

  Ensure that the destination JNDI name provided to the listener port resolves to a WebSphere MQ destination and that duplicate objects have not been defined.

- `JMSDestination mqq:`
  `com.ibm.ejs.jms.listener.MDBInvalidConfigException`

  The connection factory JNDI provided to the listener port is pointing to a connection factory that is not of type WebSphere MQ. Ensure that the connection factory JNDI name provided resolves to a WebSphere MQ connection factory and that duplicate objects have not been defined.

- `An MQ reason code is attached`

  This message implies that the listener port has at least made a connection attempt to WebSphere MQ. See Chapter 23, “JMS application with WebSphere MQ problem determination” on page 523.

19.4 Validate the solution

If you made changes to the WebSphere Application Server JMS or JNDI resources, you must restart the application server processes for changes to take effect. If changes are made to WebSphere MQ queue manager objects only, an application server restart is not normally required.

Validate the solution by checking the application server logs to ensure that the error messages no longer occur.

The presence of message WMSG0042I indicates that the listener port in question has started successfully.

20

WebSphere MQ server problem determination

WebSphere MQ server functionality was added in Version 6.1 of WebSphere Application Server. It provides another means of interoperating between WebSphere MQ and the default messaging provider in the application server.

This chapter discusses problems that can occur that are specific to the use of WebSphere MQ server.

20.1 Introduction to WebSphere MQ server configurations

WebSphere MQ server provides a means for defining the connection attributes for a client-based connection to WebSphere MQ V6 for z/OS queue managers and queue sharing groups so that a service integration bus might interconnect with them.

You can find more information about the WebSphere MQ server can be found in the following WebSphere Application Server version 6.1 Information Center topic, *Working with WebSphere MQ server* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjfp0014_.html

20.2 Verify system integrity

The first thing to check if you experience problems with a WebSphere MQ server configuration is to ensure that the WebSphere MQ server is enabled. If not, messages can appear to be "lost".

In the administrative console:

1. Select **Servers** → **WebSphere MQ servers**.
2. Select the server definition you wish to test.
3. Click **Test connection**.

If the connection is not enabled, see 20.4, "WebSphere MQ server not enabled" on page 484.

Even if the test indicates the connection is enabled, this test only checks that the deployment manager process has connectivity. It is possible that the node or server might not be configured correctly (MQ_INSTALL_ROOT problem) at run time, or that there is a firewall or network problem.

20.3 Symptoms of problems

Symptom of problems that can occur while using WebSphere MQ server include:

- Messages are not processed by WebSphere MQ server. Messages might appear to be lost.

There are no queue points for a WebSphere MQ server connection. The message transport is synchronous. The messages either go or an error will be thrown.

If the message is sent fire-and-forget, the message could end up on the exception destination for the messaging engine where the connection was made.

Specific messages that indicate this type of problem include WMSG1605E, CWSJP9999E, and WMSG1603E.

If any of these symptoms describe your problem, see 20.4, “WebSphere MQ server not enabled” on page 484.

- Unable to produce messages to a WebSphere MQ server destination.

  The application receives an exception while attempting to create a producer for a given destination or while attempting to send a message.

  Examples of messages that can accompany the exception for this type of problem include CWSIP0811W, CWSIC8002E, CWSJP0023E.

  If this is the case, see 20.5, “Unable to produce messages to a WebSphere MQ server-hosted destination” on page 486.

- Unable to consume messages from a WebSphere MQ destination.

  The application receives an exception while attempting to create a consumer for a given destination or while attempting to send a message.

  Examples of messages that can accompany the exception for this type of problem include CWSIP0815W and CWSJP0002E.

  If this is the case, see 20.6, “Unable to consume messages from a WebSphere MQ destination” on page 488.

- JMS messages arrive but appear corrupt.

  Messages arriving on WebSphere MQ destinations do not appear to have maintained standard/user specific JMS headers.

  Messages cannot be selected based on JMSXUserID or JMSCorrelationID because they have changed or contain additional spaces.

  If this is the case, see 20.7, “JMS messages arrive but appear corrupt” on page 490.

- The application is unable to connect.

  An application using a destination provided by the WebSphere MQ server connects only to the service integration bus. The application must use the correct type of connection factory. Using the wrong type of connection will produce a `JMSException`.

If you are experiencing connectivity problems:

a. Ensure that you are using a connection factory for the default messaging provider. Using the administrative console:

   i. Select **Resources** → **JMS** → ***connection factory type***

   ii. Validate the provider column for your connection factory

   Check for duplicates such as JMS resources defined with the same name for WebSphere MQ and default messaging providers.

b. Ensure the WebSphere MQ server is enabled.

If neither of these is the problem, you are probably experiencing a generic JMS connectivity issue. See Chapter 15, "JMS application problem determination" on page 401.

20.4 WebSphere MQ server not enabled

An application server hosting a messaging engine that is used to connect to the WebSphere MQ server must have access to the required minimum level of the WebSphere MQ client.

A reference to the WebSphere MQ client code is provided by the WebSphere Application Server environment variable MQ_INSTALL_ROOT, and is defined at the `node` scope. By default, this variable points to the client that is provided with the application server, and in most cases this client is sufficient. However, you might need to modify this if your configuration is to run in `bindings` mode.

If the MQ_INSTALL_ROOT variable references the wrong location or the referenced WebSphere MQ client is not a supported level, then the WebSphere MQ server functionality is disabled.

20.4.1 Analyze SystemOut for the messaging engine

Examine the application server log files for error messages prefixed with CWSJP and WMSG.

The following list contains specific error messages addressed by this chapter. If you find an error message other than those listed and the information included in the messages is not sufficient to determine the root cause of the message, see Chapter 27, "The next step" on page 581:

- If you see this message, see 20.4.2, “MQ JMS client not detected” on page 485.

  ```
  WMSG1605E: It was not possible to detect the MQ JMS client at the
  specified path {0}.
  ```

- If you see this message, see 20.4.3, “WebSphere MQ client version not supported” on page 485.

  ```
  CWSJP9999E: WebSphere MQ Server functionality cannot operate with
  the level of WebSphere MQ client found MQ_Client_Level_Found
  WebSphere MQ Server functionality has been disabled.
  ```

- If you see this message, see 20.4.4, “Previous version registered” on page 486.

  ```
  WMSG1603E: An internal error occurred. It was not possible to
  register the WebSphere MQ JMS client with the application server due
  to exception {0}.
  ```

20.4.2 MQ JMS client not detected

The symptom of this problem is the following message:

```
WMSG1605E: It was not possible to detect the MQ JMS client at the
specified path {0}.
```

Ensure that the WebSphere Application Server MQ_INSTALL_ROOT variable is set correctly and that the WebSphere Application Server process has the correct access permissions.

20.4.3 WebSphere MQ client version not supported

The symptom of this problem is this message:

```
CWSJP9999E: The version of WebSphere MQ Client referenced by the
WebSphere Application Sever MQ_INSTALL_ROOT variable does not meet the
minimum supported level of WebSphere MQ.
```

Check the system requirements for WebSphere Application Server 6.1 to ensure that your combination of WebSphere Application Server and WebSphere MQ versions is supported for your environment.

See *System Requirements for WebSphere Application Server V6.1 at:*

http://www.ibm.com/support/docview.wss?rs=180&uid=swg27007651

If it is not possible to apply the specified maintenance then you should restore the MQ_INSTALL_ROOT variable to its original setting of ${WAS_INSTALL_ROOT}/lib/WMQ.

20.4.4 Previous version registered

The symptom of this problem is this message:

```
WMSG1603E: An internal error occurred. It was not possible to register
the WebSphere MQ JMS client with the application server due to
exception {0}.
```

A problem has occurred when loading the client because a previous version was already registered. This generally occurs when changing between versions of the client.

To resolve this problem:

1. Ensure no processes are running for the given WebSphere Application Server profile.
2. Run the **osgiCfgInit** command from the bin directory of the given WebSphere Application Server profile.
3. Restart the WebSphere Application Server nodes and servers for the given profile.

20.4.5 Validate the solution

If you changed the WebSphere MQ_INSTALL_ROOT variable or modified access permissions, restart each application server that hosts a messaging engine that interacts with the WebSphere MQ server.

After the restart, check the application server logs to ensure no error messages prefixed CWSJP or WMSG are reported.

20.5 Unable to produce messages to a WebSphere MQ server-hosted destination

This topic examines common problems encountered when attempting to produce messages to a destination hosted on a WebSphere MQ server.

20.5.1 Collect diagnostics

In addition to gathering information from the application log and the SystemOut messaging engine (see 10.3, "Collect diagnostics" on page 339), you need to collect the information from the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.

20.5.2 Analyze the application log

First, examine the exception in the application log; specifically, look at the final **Caused by:** section. The error will look similar to Example 20-1.

Example 20-1 Errors indicating problem producing messages

```
Caused by: com.ibm.wsspi.sib.core.exception.SISessionDroppedException:
CWSIP0811W: A message could not be sent to destination MQQ assigned to
WebSphere MQ Server bus member QIU0-ghbus.  The WebSphere MQ completion
code was 2.  The WebSphere MQ reason code was 2053.
        at
com.ibm.ws.sib.comms.common.JFAPCommunicator.parseSingleException(JFAPC
ommunicator.java:1874)
        ... 55 more
Caused by: com.ibm.websphere.sib.exception.SIErrorException:
CWSIC8002E: An internal error occurred. An unknown or unexpected
exception was thrown by the core A
PI: exception
com.ibm.ws.sib.remote.mq.exceptions.CorruptRMQSessionException:
CWSJP0023E: A Websphere MQ error was encountered when using the
Websphere MQ queue GHQ on the Websphere MQ queue manager Name :QIU0,
Host :winmvsu0, Port :1481, Channel :MQClient, Bus :ghbus, BM Name
:QIU0-ghbus, BM Uuid :FDA1D736CCFAB7FC, Auth alias :, Transport
:OutboundBasicWMQClient: com.ibm.mq.MQException: MQJE001: Completion
Code 2, Reason 2053.
        at
com.ibm.ws.sib.comms.common.JFAPCommunicator.parseSingleException(JFAPC
ommunicator.java:1980)
        ... 55 more
Caused by: com.ibm.websphere.sib.exception.SIErrorException:
CWSIC8002E: An internal error occurred. An unknown or unexpected
exception was thrown by the core A
PI: exception com.ibm.mq.MQException: MQJE001: Completion Code 2,
Reason 2053.
```

Examine the exception information and the nested exception information contained in the **Caused by** sections:

- If a WebSphere MQ reason code is given as the root cause, then the failure is likely be the result of a WebSphere MQ problem (as in the case above).

  If this is the case, see Chapter 23, "JMS application with WebSphere MQ problem determination" on page 523.

- Otherwise, the problem is likely to reside within the service integration bus.

  If this is the case, see Chapter 20, "WebSphere MQ server problem determination" on page 481. Specifically, look at section 15.5, "JMS application unable to produce messages" on page 417.

20.6 Unable to consume messages from a WebSphere MQ destination

The JMS application might be unable to consume messages from a destination hosted on a WebSphere MQ server.

This topic examines common problems encountered when attempting to consume messages from a destination hosted on a WebSphere MQ server.

20.6.1 Collect diagnostics

In addition to the application log and messaging engine SystemOut already collected in 10.3, "Collect diagnostics" on page 339, look in the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.

20.6.2 Analyze diagnostics

Search for and examine the JMSException for indications of the problem. Specifically, examine the message included in the final **Caused by:** section. It will look something like Example 20-2.

Example 20-2 Stack trace in application log containing the exception

```
Stack Dump =
com.ibm.ws.sib.processor.impl.exceptions.RMQSessionDroppedException:
CWSIP0815W: A message could not be received from destination MQQ
assigned to WebSphere MQ Server bus member QIUO-ghbus. The WebSphere MQ
completion code was 2. The WebSphere MQ reason code was 2016.
```

```
   at
com.ibm.ws.sib.processor.impl.mqproxy.RMQCursor.next(RMQCursor.java:208
)
   at
com.ibm.ws.sib.processor.impl.RMQAsynchThread$AsynchRunnable.runAsynchC
onsumer(RMQAsynchThread.java:553)
   at
com.ibm.ws.sib.processor.impl.RMQAsynchThread$AsynchRunnable.run(RMQAsy
nchThread.java:429)
   at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1469)
Caused by:
com.ibm.ws.sib.remote.mq.exceptions.CorruptRMQSessionException:
CWSJP0002E: An internal messaging error occurred in
com.ibm.ws.sib.remote.mq.impl.AbstractRMQSession.processException,
1:952:1.54 due to an exception com.ibm.mq.MQException: MQJE001:
Completion Code 2, Reason 2016
   at
com.ibm.ws.sib.remote.mq.impl.AbstractRMQSession.processSessionExceptio
n(AbstractRMQSession.java:957)
   at
com.ibm.ws.sib.remote.mq.impl.AbstractRMQSession.processSessionExceptio
n(AbstractRMQSession.java:897)
   at
com.ibm.ws.sib.remote.mq.impl.BaseCursor.next(BaseCursor.java:223)
   at
com.ibm.ws.sib.processor.impl.mqproxy.RMQCursor.next(RMQCursor.java:135
)
   ... 3 more
Caused by: com.ibm.mq.MQException: MQJE001: Completion Code 2, Reason
2016
   at com.ibm.mq.MQQueue.getMsg2(MQQueue.java:1071)
```

Examine the exception and nested exception information contained in the **Caused by** sections:

- If a WebSphere MQ reason code is given as the root cause, then the failure is likely the result of a WebSphere MQ problem (as in the case above).

  If this is the case, see 23.5, “JMS application is unable to consume messages” on page 527.

- Otherwise, the problem is likely to reside within the service integration bus.

  If this is the case, see Chapter 20, “WebSphere MQ server problem determination” on page 481. Specifically, look at section 15.7, “JMS application unable to consume messages” on page 423.

20.7 JMS messages arrive but appear corrupt

This topic examines the reasons that messages arrive at the intended destination but appear to be missing information or have incorrect data in them.

20.7.1 Collect diagnostics

Locate the header and payload information of the message that appears incorrect.

Work with the WebSphere MQ administrator to find and examine the message.

20.7.2 Analyze diagnostics

Examine the message headers and payload to determine if the message is incorrect:

- If the message is missing custom JMS headers, see 20.7.3, “WebSphere MQ RFH2 headers not enabled” on page 490.
- If the JMSXUserID field is either padded with spaces or has been truncated, see 20.8, “JMSXUserID field padded with spaces or truncated” on page 491.
- If the JMSCorrelationID field is padded with spaces or truncated, see 20.8.1, “JMSCorrelationID field padded with spaces or truncated” on page 492.

20.7.3 WebSphere MQ RFH2 headers not enabled

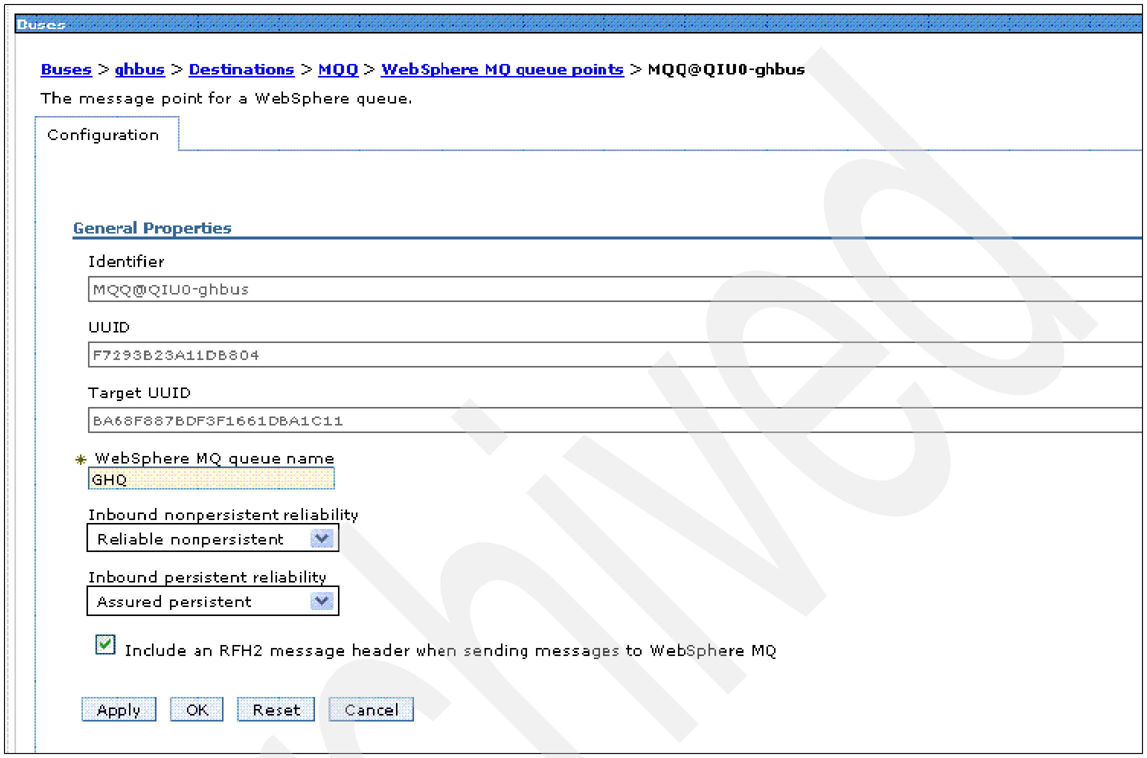
If the custom JMS headers are not propagated, the likely reason for this is that the definition of the destination within the service integration bus does not have the WebSphere MQ RFH2 headers enabled.

If you are expecting messages to arrive on WebSphere MQ queues with these headers included, then you must ensure that the **Include an RFH2 message header** property is checked for the WebSphere MQ queue point you are using.

In the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Select **Destinations.**
4. Click the destination configured to use the WebSphere MQ server.
5. Select **queue points**.

6. Click on the name of the WebSphere MQ queue point.
7. Select the **Include an RFH2 message header when sending messages to WebSphere MQ** check box (as illustrated in Figure 20-1).


Buses

Buses > ghbus > Destinations > MQQ > WebSphere MQ queue points > MQQ@QIU0-ghbus

The message point for a WebSphere queue.

Configuration

General Properties

Identifier: MQQ@QIU0-ghbus

UUID: F7293B23A11DB804

Target UUID: BA68F887BDF3F1661DBA1C11

* WebSphere MQ queue name: GHQ

Inbound nonpersistent reliability: Reliable nonpersistent

Inbound persistent reliability: Assured persistent

[✓] Include an RFH2 message header when sending messages to WebSphere MQ

Figure 20-1 Setting to include an RFH2 message header

20.8 JMSXUserID field padded with spaces or truncated

If the JMSXUserID field is either padded with spaces or has been truncated, the MQMD is restricted to 8 bytes; therefore, a maximum of 8 characters can be used.

If the field is padded with spaces, then this is because the current implementation copies the user ID from an MQ message when it brings the message into the service integration bus. A direct copy of the 8 byte field is performed so that the user ID appears in the service integration bus as 8 characters. If the field was originally shorter, then it is padded with spaces. The same is true for messages flowing from the service integration bus to WebSphere MQ.

This behavior is a limitation in the current release; therefore, you need to consider it when you develop applications that use this field.

20.8.1 JMSCorrelationID field padded with spaces or truncated

The JMSCorrelationID is processed in much the same way as the JMSXUserID (described above). The field is made to fit the size provided; if the field is shorter than the field length it is padded with 0s. This behavior is a limitation in the current release; therefore, you need to consider it when you develop applications that use this field.

20.8.2 Validate the solution

If you made changes to the WebSphere MQ server definition or JNDI resources, you must restart the application servers that interact with the WebSphere MQ server for the changes to take effect.

Send messages and ensure the expected message properties are now correctly propagated.

21

WebSphere MQ configuration

This chapter discusses problem determination techniques and possible root causes for problems when using the WebSphere MQ messaging provider.

21.1 Identify symptoms

When applications running in an application server use the WebSphere MQ messaging provider, the server must have access to the WebSphere MQ client code. These symptoms are an indication that the WebSphere MQ messaging provider is not enabled:

- JMS objects such as queue connection factories are defined, but do not appear in JNDI.
- WMSG*xxxx*E error messages appear in the system logs indicating that WebSphere MQ JMS binders have been disabled.

If your application is running in an application server, see 21.2, “WebSphere MQ messaging provider not enabled” on page 494.

WebSphere MQ JMS applications running outside of the application server require access to the WebSphere MQ client binaries. These symptoms are an indication that the WebSphere messaging client cannot resolve the WebSphere MQ client binaries:

- This exception is reported when running client application:

  ```
  java.lang.Exception: De-reference of JMS provider's Reference failed
  - check provider is on classpath
  ```

- Client application gets JMSExceptions when attempting to run in bindings mode with a stack trace containing the WebSphere MQ return code 2046.

If your application is running outside of an application server, see 21.3, “Messaging client fails to resolve the WebSphere MQ libraries” on page 497.

21.2 WebSphere MQ messaging provider not enabled

For WebSphere Application Server to utilize WebSphere MQ as a messaging provider, it must access the WebSphere MQ JMS client code. This is resolved as part of the WebSphere Application Server install process. The WebSphere MQ code is located under *app_server_root*/lib/WMQ and is referenced by the WebSphere Variable MQ_INSTALL_ROOT at the node scope.

Depending on your WebSphere Application Server configuration and WebSphere MQ infrastructure, it might become necessary to change the MQ_INSTALL_ROOT variable. The most common reasons for this are:

- To point to a different version of the WebSphere MQ client

- To utilize WebSphere MQ queue managers running on the same hardware as the WebSphere Application Server environment, that is, run in bindings mode.

If the install directory is set incorrectly, or there are permission problems on the target directory, then the WebSphere MQ JMS binders will be disabled preventing the use of WebSphere MQ as a JMS provider.

21.2.1 Analyze SystemOut

Examine the SystemOut log for the server running the application. Look for error messages prefixed with WMSG:

- If an error was encountered during the loading the WebSphere MQ Client, you see this message in the log:

  ```
  WMSG0902E: The WebSphere MQ JMS Binders have been disabled as either
  the WebSphere MQ Client has not been installed, or the
  MQ_INSTALL_ROOT variable has not been set.
  ```

- If you see this message, see 21.2.2, "MQ JMS client not detected" on page 495.

  ```
  WMSG1605E: It was not possible to detect the MQ JMS client at the
  specified path {0}.
  ```

- If you see this message, see 21.2.3, "WebSphere MQ Client version not supported" on page 496.

  ```
  WMSG1604E: It was not possible to use the specified WebSphere MQ JMS
  client at path {0} because it is at version {1}, the minimum
  supported version is {2}.
  ```

- If you see this message, see 21.2.4, "Previous version registered" on page 496.

  ```
  WMSG1603E: An internal error occurred. It was not possible to
  register the WebSphere MQ JMS client with the application server due
  to exception {0}.
  ```

21.2.2 MQ JMS client not detected

The symptom of this problem is this message:

```
WMSG1605E: It was not possible to detect the MQ JMS client at the
specified path {0}.
```

Ensure that the WebSphere Application Server MQ_INSTALL_ROOT variable is set correctly and that the WebSphere Application Server process has the correct access permissions.

21.2.3 WebSphere MQ Client version not supported

The symptom of this problem is this message:

```
WMSG1604E: It was not possible to use the specified WebSphere MQ JMS
client at path {0} because it is at version {1}, the minimum supported
version is {2}.
```

The version of WebSphere MQ Client referenced by the WebSphere Application Sever MQ_INSTALL_ROOT variable does not meet the minimum supported level of WebSphere MQ.

Check the System Requirements for WebSphere Application Server 6.1 to ensure that your combination of WebSphere Application Server and WebSphere MQ versions is supported for your environment:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007651

21.2.4 Previous version registered

The symptom of this problem is this message:

```
WMSG1603E: An internal error occurred. It was not possible to register
the WebSphere MQ JMS client with the application server due to
exception {0}.
```

A problem has occurred when loading the client as a previous version was already registered. This generally occurs when changing between versions of the client, to resolve this problem perform this steps:

1. Ensure no processes are running for the given WebSphere Application Server profile.
2. Run the **osgiCfgInit** command from the bin directory of the given WebSphere Application Server profile.
3. Restart the WebSphere Application Server nodes and servers for the given profile.

21.2.5 Validate the solution

Whenever the WebSphere MQ_INSTALL_ROOT variable is changed or access permissions modified, it will be necessary to restart the application server processes for the changes to take effect. After the restart, check the application server logs to ensure no error messages prefixed WMSG are reported.

21.3 Messaging client fails to resolve the WebSphere MQ libraries

WebSphere MQ JMS applications running outside of the application server require access to the WebSphere MQ client binaries. For the WebSphere Application Server client container, the location of the WebSphere MQ client is specified within the ${WAS_INSTALL_ROOT}/bin/launchClient script.

21.3.1 Examine the application messages

Problems will be reported by the J2SE application client. Messages are written to the prompt where the launchClient command was run.

Examine the application messages for exceptions. Look for these messages:

- `java.lang.Exception: De-reference of JMS provider's Reference failed - check provider is on classpath`
- Exceptions when attempting to run in **`bindings`** mode with a stack trace that contains the WebSphere MQ return code `2046`.

These messages indicate that the WebSphere messaging client cannot resolve the WebSphere MQ client binaries. If you are experiencing these symptoms, open the ${*WAS_INSTALL_ROOT*}/bin/launchClient script in an editor, and check that the JMS_PATH references the correct location for the WebSphere MQ JMS jar files to be used:

- If the JMS_PATH is referencing the incorrect location, update the path to point to the correct location of the MQ JMS jar files that you want to use.
- If the JMS_PATH is referencing the correct location, but there is still a problem, the most likely cause is that the user launching the application does not have the correct access permissions. Update the user permissions and rerun the client application.

It is not necessary to restart the application server for changes to the ${*WAS_INSTALL_ROOT*}/bin/launchClient script to take effect.

22

WebSphere MQ link problem determination

This chapter identifies problem determination activities for the WebSphere MQ link component of the default messaging provider in WebSphere Application Server V6.1.

22.1 Introduction to WebSphere MQ link

A WebSphere MQ link enables a service integration bus to interconnect with a WebSphere MQ queue manager. Depending on the chosen configuration, the WebSphere MQ link can support both point-to-point and publish-subscribe messaging paradigms.

For a detailed description of the WebSphere MQ link and its configuration options, see *How the WebSphere MQ link interoperates with WebSphere MQ* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.express.doc/concepts/cjc1002_.html

Common problems with the WebSphere MQ link can categorized into these three areas:

- Sender channels fail to start.
- Message flow problems.
- Corrupt messages.

22.2 Verify integrity

The first step in diagnosing the problem is to do this verify the integrity of the system:

- Verify the WebSphere MQ link is running.
- Verify the WebSphere MQ link sender channel is running.
- Verify the WebSphere MQ link receiver channel is running.

For information about the various states of the MQ link, the states of the sender and receiver channels, and their meanings, see *States of the WebSphere MQ link and its channels* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/tasks/tjc0003_.html

22.2.1 Verify the WebSphere MQ link is running

Check the state of the WebSphere MQ link with the administrative console as follows:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.

2. Click **Messaging engines.** Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**.
4. Check the Status column.

   A `Running` status indicates a network connection has been established between the application server and queue manager. Messages will be transmitted.

   A `Standby` status indicates the sender channel is not network-connected to its WebSphere MQ counterpart receiver channel. It is waiting for a message to send before attempting to establish a connection.

   If the link is in a `Stopped` state, attempt to start it by selecting the link and clicking **Start**.

22.2.2 Verify the WebSphere MQ link sender channel is running

Check the state of the sender channel with the administrative console as follows:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines.** Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check
4. Select the **Sender Channel**.
5. Check the Status column.

   If the channel is in `Stopped` state, attempt to start it. If it is in `Retry` state, work with the WebSphere MQ administrator to ensure the WebSphere MQ resources are available.

22.2.3 Verify the WebSphere MQ link receiver channel is running

To view or modify the receiver channel status, in the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines.** Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check.

4. Select **Receiver Channel**.
5. Check the Status column.

   If the channel is in `Stopped` state, attempt to start it.

22.3 Identify symptoms

Find the symptoms that most closely resembles the symptoms you are experiencing:

- The sender channel from WebSphere MQ link fails to start:
  - The sender channel is in state `Retry`.
  - Attempts to start the WebSphere MQ link sender channel manually result in error similar to this:

    ```
    CWSIQ0017E: The sender channel Sender_Channel_Name for MQLink
    MQLink_Name has failed to establish a connection with the remote
    host Host_Name/IP_Address because the remote listener for port
    Port_Number is not available.
    ```
  - Messages that are targeted at the WebSphere MQ queue manager are backing up on the WebSphere MQ link sender channel.

  If you are experiencing similar symptoms, see 22.4, “Sender channel fails to start” on page 503.
- Receiver channel from WebSphere MQ fails to start.
  - Attempts to manually start the WebSphere MQ sender channel result in the channel going into a `Retry` state.
  - Messages that are targeted at the service integration bus start to back up on the WebSphere MQ sender channel.

  If you are experiencing similar symptoms, see 22.5, “Receiver channel fails to start” on page 507.
- Message flow problem from the service integration bus to WebSphere MQ:
  - *No message flow:* JMS messages are sent successfully but do not arrive on the WebSphere MQ destination. Examination of the appropriate channels confirms they are in a `Running` state, but that the message count is not incrementing.
  - *Messages apparently lost:* JMS messages are sent successfully but do not arrive on the WebSphere MQ destination. Examination of the appropriate channels confirms they are in a `Running` state and that the message count for the channel shows that messages have successfully flowed across the channel.

If you are experiencing similar symptoms, see 22.6, "Encountering a message flow problem from the service integration bus to WebSphere MQ" on page 509.

- Message flow problem from WebSphere MQ to the service integration bus:
  - *No message flow*: JMS messages are sent successfully but do not arrive on the service integration bus destination. Examination of the appropriate channels confirms they are in `Running` state, but that the message count is not incrementing.
  - *Messages lost*: JMS messages are sent successfully but do not arrive on the service integration bus destination. Examination of the appropriate channels confirms they `Running` state and that the message count for the channel shows that messages have successfully flowed across the channel.

  If you are experiencing similar symptoms, see 22.7, "Encountering a message flow problem from WebSphere MQ to the service integration bus" on page 515.

- Messages flow across the WebSphere link but appear corrupt.

  Messages arriving on WebSphere MQ destinations do not appear to have maintained standard or user-specific JMS headers.

  If you are experiencing similar symptoms, see 22.8, "Messages flow across the WebSphere link but appear corrupt" on page 518.

22.4 Sender channel fails to start

This topic examines some of the common configuration issues that prevent the WebSphere MQ link sender channel from starting. The sender channel is required to enable communication from a service integration bus to WebSphere MQ.

22.4.1 Collect diagnostics

In addition to the SystemOut log for the application server hosting the messaging engine configured with the WebSphere MQ link (see 10.3, "Collect diagnostics" on page 339), locate the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.

22.4.2 Analyze SystemOut

Examine the application server log files for the server hosting the messaging engine configured with the WebSphere MQ link. Look for error messages with prefixes CWSIC and CWSIQ:

- If you see this message, see 22.4.3, “Cannot contact listener” on page 504.

  ```
  CWSIQ0017E: The sender channel {0} for MQLink {1} has failed to
  establish a connection with the remote host {2} because the remote
  listener for port {3} is not available.
  ```

- If you see this message, see 22.4.4, “Channel not available” on page 505.

  ```
  CWSIC3108E: The WebSphere MQ link {0} ended because channel {1} is
  not currently available on the remote system.
  ```

- If you see this message, see 22.4.5, “Attempt to send message failed” on page 505.

  ```
  CWSIC3080E: The WebSphere MQ link {0} has ended because the remote
  queue manager {1} cannot receive a message.
  ```

- If you see this message, see 22.4.6, “Message sequence number error” on page 506.

  ```
  CWSIC3011E: The WebSphere MQ link {0} and the remote queue manager
  do not agree on the next message sequence number. A message with
  sequence number {1} has been sent when sequence number {2} was
  expected.
  ```

22.4.3 Cannot contact listener

The symptom for this problem is this message:

```
CWSIQ0017E: The sender channel {0} for MQLink {1} has failed to
establish a connection with the remote host {2} because the remote
listener for port {3} is not available.
```

Error message CWSIQ0017E indicates that it is not possible to connect to the listener on the host and port provided.

To resolve the problem:

- Ensure that the host and port displayed in the error message are correct. Modify them in the WebSphere MQ link sender channel as required.
- Work with your WebSphere MQ administrator to ensure that an associated listener is running on the specified port.
- Verify that the connection attempt is not blocked by a firewall.

22.4.4 Channel not available

The symptom for this problem is this message:

```
CWSIC3108E: The WebSphere MQ link {0} ended because channel {1} is not
currently available on the remote system.
```

Error message CWSIC3108E indicates that the channel name specified is not available on the WebSphere MQ queue manager.

To resolve the problem:

- Verify that the channel name is correct (and uses the correct case).
- Work with your WebSphere MQ administrator to ensure that the receiver channel is defined and in `Started` state on the remote queue manager.

If these steps fail to resolve the problem, examine the WebSphere MQ queue manager logs for error codes that give a clearer indication of the problem. The most common MQ error codes are:

- AMQ9534: Channel *Channel_Name* is currently not enabled.

  This message indicates that the channel has been placed in `Stopped` state.

  Start the channel manually from WebSphere MQ to allow communication to begin. Investigate why the channel is stopped.
- AMQ9519: Channel *Channel_Name* not found

  This message indicates that the channel has not been defined or that the channel name entered within the WebSphere MQ link sender is incorrect.

  Define the required receiver channel or adjust the channel name provided for the WebSphere MQ link sender accordingly.

22.4.5 Attempt to send message failed

The symptom for this problem is this message:

```
CWSIC3080E: The WebSphere MQ link {0} has ended because the remote
queue manager {1} cannot receive a message.
```

CWSIC3080E indicates that an attempt to send a message to WebSphere MQ failed.

This problem can occur when a message is sent to WebSphere MQ but WebSphere MQ could not forward the message to the requested destination. In addition, it could not place the message on the queue managers dead-letter-queue. The channel is in doubt about which messages have been committed by WebSphere MQ for the unit of work that it has sent. The `In doubt` setting on the channel is set to `true`, which prevents WebSphere Application Server from sending any more messages to WebSphere MQ until it is resolved. The message is retained on the WebSphere MQ link sender channel. So see the `In doubt` value, in the administrative console, look on the Runtime tab of the sender channel.

To diagnose this problem, work with the WebSphere MQ administrator. The diagnosis process is similar to the one you would use if messages could not be sent from WebSphere MQ to WebSphere Application Server. For more information, see 22.7, "Encountering a message flow problem from WebSphere MQ to the service integration bus" on page 515.

22.4.6 Message sequence number error

The symptom for this problem is this message:

```
CWSIC3011E: The WebSphere MQ link {0} and the remote queue manager do
not agree on the next message sequence number. A message with sequence
number {1} has been sent when sequence number {2} was expected.
```

Error message CWSIC3011E indicates that, for some reason, the two ends of the channel do not agree on the next message sequence number.

This problem might be the result of deleting and recreating one end of the channel pairing. If this is the case, reset the sequence number on the WebSphere MQ link sender channel.

To check the state of the WebSphere MQ link sender channel with the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines.** Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check.
4. Select the **Sender Channel.**
5. Select the box next to the sender channel, and click **Reset**.

Note: If you are using WebSphere MQ NPM_SPEED settings, message CWSIC3011E might also occur as a result of a WebSphere Application Server defect. This problem is reported in APAR PK49927.

22.4.7 Validate the solution

If you made changes to the WebSphere MQ link sender channel definition, restart the application server configuration so that they take effect.

Start the sender channel and ensure that the channel is in the `Running` state.

22.5 Receiver channel fails to start

This topic examines some of the common configuration issues that can be encountered when attempting to start WebSphere MQ sender channels to enable communication from WebSphere MQ to a service integration bus.

22.5.1 Collect diagnostics

Locate the WebSphere MQ queue manager error logs. Work with your WebSphere MQ administrator to collect the appropriate logs for analysis for your platform.

22.5.2 Analyze the WebSphere MQ queue manager logs

Examine the WebSphere MQ queue manager error log files and check for message codes: AMQ9520, AMQ9558, and AMQ9202:

- If you see this message, see 22.5.3, “Channel not defined remotely” on page 507.

  `AMQ9520:` `Channel not defined remotely`
- If you see this message, see 22.5.4, “Remote channel not available” on page 508.

  `AMQ9558:` `Remote Channel is not currently available`
- If you see this message, see 22.5.5, “Remote host not available” on page 508.

  `AMQ9202:` `Remote host Host Name not available, retry later`

22.5.3 Channel not defined remotely

The symptom for this problem is this message:

`AMQ9520:` `Channel not defined remotely`

This message indicates that the WebSphere MQ link channel name, defined on the WebSphere MQ sender, cannot be located on the remote (WebSphere Application Server) system.

Ensure the channel name is correct, observing any capitalization, and that the appropriately named WebSphere MQ link receiver channel is defined for the WebSphere MQ link configuration.

22.5.4 Remote channel not available

The symptom for this problem is this message:

`AMQ9558:` `Remote Channel is not currently available`

This message indicates that the channel on the remote (WebSphere Application Server) system is defined but is not available. This problem might be due to the channel being placed in the `Stopped` state or because insufficient resources are available.

Ensure the WebSphere MQ link receiver is not in `Stopped` state. If it is, start it.

If the channel is not stopped, check the application server logs for further indications of the reason why the channel is unavailable.

22.5.5 Remote host not available

The symptom for this problem is this message:

`AMQ9202:` `Remote host Host Name not available, retry later`

This message indicates that it was not possible to connect to the service integration bus on the host and port provided under the connection name (conname) attribute of the WebSphere MQ sender channel definition.

Verify that the host and port are correct. The port number should match the port defined for the SIB_MQ_ENDPOINT_ADDRESS on the appropriate application server. The default for this port is `5558`.

To check the ports for the application servers in your configuration use the administrative console to:

1. Select **Servers** → **Application Servers**.
2. Click the server name on which your WebSphere MQ link is defined, to open the details page.
3. Click to expand the Ports sub-section under the Communications heading.

4. Note the SIB_MQ_ENDPOINT_ADDRESS port number.

22.5.6 Validate the solution

If you made changes to the WebSphere MQ link definition or JNDI resources, restart the application server so that they take effect.

Ask the WebSphere MQ administrator to start the WebSphere MQ sender channel.

22.6 Encountering a message flow problem from the service integration bus to WebSphere MQ

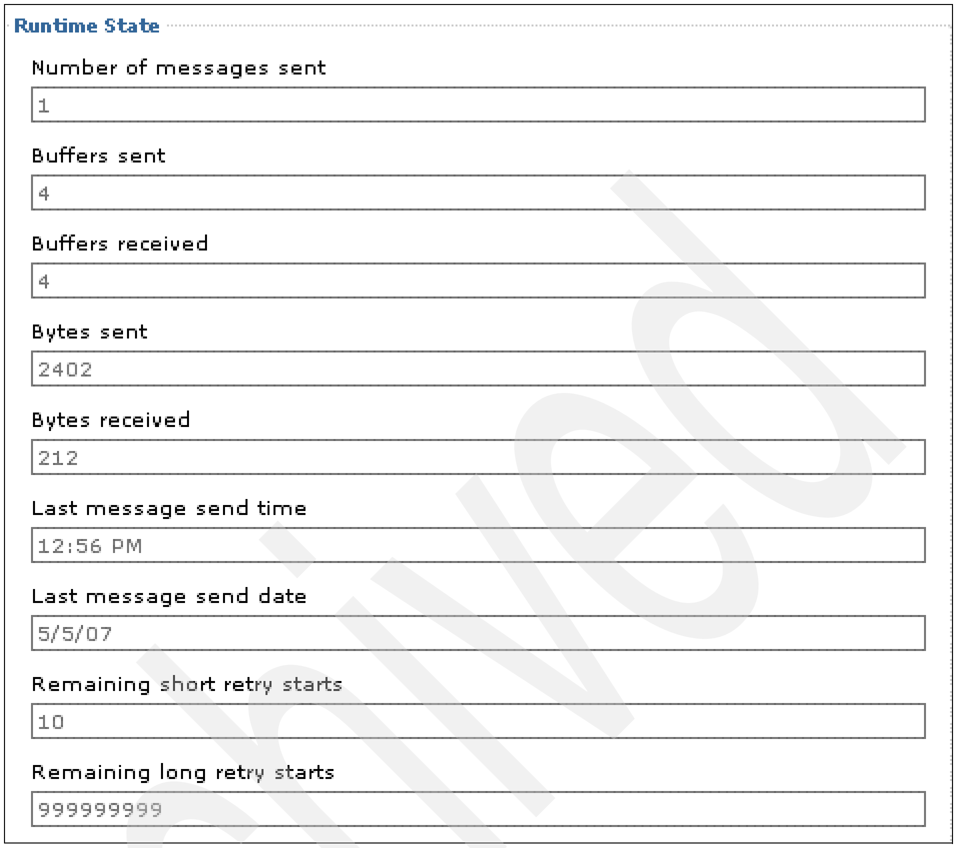
This topic examines some of the issues that can affect message flow from the service integration bus using a WebSphere MQ link to WebSphere MQ destinations.

22.6.1 Determine if the message has reached the WebSphere MQ link

Determine how far through the system the message has progressed. Using the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines.** Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you wish to check.
4. Select **Sender Channel**.
5. Click on the name of the WebSphere MQ sender channel.
6. Select the **Runtime** tab.

You see the panel that displays the runtime state of the sender channel, as illustrated in Figure 22-1 on page 510.


Runtime State

Number of messages sent
1

Buffers sent
4

Buffers received
4

Bytes sent
2402

Bytes received
212

Last message send time
12:56 PM

Last message send date
5/5/07

Remaining short retry starts
10

Remaining long retry starts
999999999

Figure 22-1 Sender channel runtime state

Check the **Number of messages sent** field. This field should increment when messages are sent.

22.6.2 Collect diagnostics

In addition to the SystemOut log for the server hosting the messaging engine configured with the WebSphere MQ link, and the application log, you need the following information:

- Examine the exception destination for the messaging engine hosting the MQ link. For more information about exception destinations, see *Exception destinations* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/concepts/cjo0004_.html

Note: If the **number of messages sent** count on the sender channel is not incrementing, the messages are not reaching WebSphere MQ and you do not need diagnostic data from WebSphere MQ.

- Locate the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.
- Examine the dead letter queue on the appropriate queue managers on the WebSphere MQ system.

22.6.3 Analyze diagnostics

If the messages on the sender channel are *incrementing*, the problem is occurring on the WebSphere MQ side. See 22.6.5, “WebSphere MQ errors” on page 513.

If the messages on the sender channel are *not incrementing*, the messages are not making it as far as the WebSphere MQ link. Continue analyzing the diagnostics.

If you are producing the messages using a messaging engine that is not hosting the MQ link, you could be experiencing a problem related to inter- messaging engine communication. See Chapter 16, “Messaging in a multiple messaging engine environment” on page 429.

Analyze SystemOut

Search the SystemOut log for:

```
CWSIC3096I: While sending message to queue manager Queue_Manager_Name
down WebSphere MQ link MQLink_Name one or more messages were put to the
exception destination.
```

This message indicates that the message could not be sent across the MQ Link and that the message has been placed on the exception destination.

In general, this error message is accompanied by another error message indicating the cause, which is also added into the message when routed to the exception destination.

If this message is also present, see 22.6.4, “Message is too large” on page 512.

```
CWSIK0104E: The message is too big to be handled by remote MQ; sending
to exception destination: message size {0}, maximum message size {1}.
```

Analyze the application log

Examine the application log for exception information that might help you determine the cause. The initial exception received by the application (deployed server side) resides here. This exception should help narrow the JMS resources involved in the operation which failed.

22.6.4 Message is too large

The symptom for this problem is this message:

```
CWSIK0104E: The message is too big to be handled by remote MQ; sending
to exception destination: message size {0}, maximum message size {1}.
```

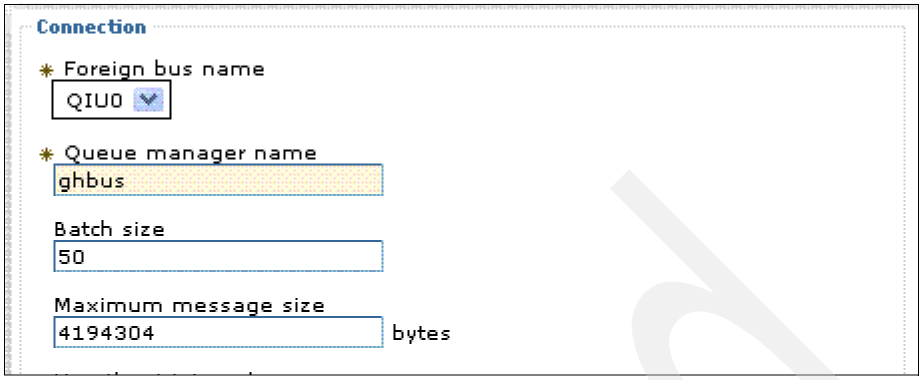
A common reason a message is routed to the exception destination when sending to WebSphere MQ is because the message is to large to be processed by the MQ system. It is bigger than allowed by the maximum message size negotiated by the sender/receiver channel pairing.

You can increase this size can be increased for larger messages. However, you must adjust the size on both sides of the channel.

In WebSphere Application Server, the maximum message size setting is configured against the WebSphere MQ link instead of on the channel. To configure this setting, from the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines.** Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link you want to check.

   You see the WebSphere MQ link configuration panel, as illustrated in Figure 22-2 on page 513.


Connection

Foreign bus name: QIU0

Queue manager name: ghbus

Batch size: 50

Maximum message size: 4194304 bytes

Figure 22-2 Adjusting the maximum message size

To change the maximum message size on WebSphere MQ, work with your WebSphere MQ administrator to update the WebSphere MQ channel configuration.

22.6.5 WebSphere MQ errors

If the sender channel number of messages sent is incrementing, messages are successfully making it to the WebSphere MQ link.

Work with your WebSphere MQ administrator to check the appropriate queue manager's dead letter queue and examine the WebSphere MQ error logs to determine where the message has been routed.

In general, if the message is persistent, it will either be on a transmission queue pending a send to another queue manager as part of a multi-hop configuration, or it will have been placed upon the dead letter queue.

If the message has been placed on the dead letter queue, check the dead letter header for an indication of the reason the message was placed there.

Common reasons are as follows:

- `MQRC_Q_FULL`

  The target destination has already reached its maximum capacity and cannot receive any more messages.

  Investigate why the WebSphere MQ queue has become full and, if appropriate, increase the capacity of the queue.

- `MQRC_UNKNOWN_OBJECT_NAME`

The queue name specified is not defined in the local WebSphere MQ queue manager. The destination that was used is included as part of the dead letter header information.

Ensure that it is correct. Either adjust the JMS/alias definition for the destination in the originating application server configuration, or define the queue in the target queue manager.

- `MQRC_UNKNOWN_REMOTE_Q_MGR`

  The message was sent specifying a target queue manager that is not known by the receiving queue manager.

  The primary reason this problem occurs is that the foreign bus does not have the same name as the remote queue manager; for example, the queue manager name is `QUI1` but the foreign bus is named `MQSystem`. If this is the case, messages that flow from the service integration bus to the queue manager will have the queue manager name set to `MQSystem` which is not known by the receiving queue manager.

  To overcome this problem, use an `@` notation in the queue name specified in the JMS queue definition to define the specific target queue manager. See the example shown in Figure 22-3.

  The format is:

  queue_name@queue_manager

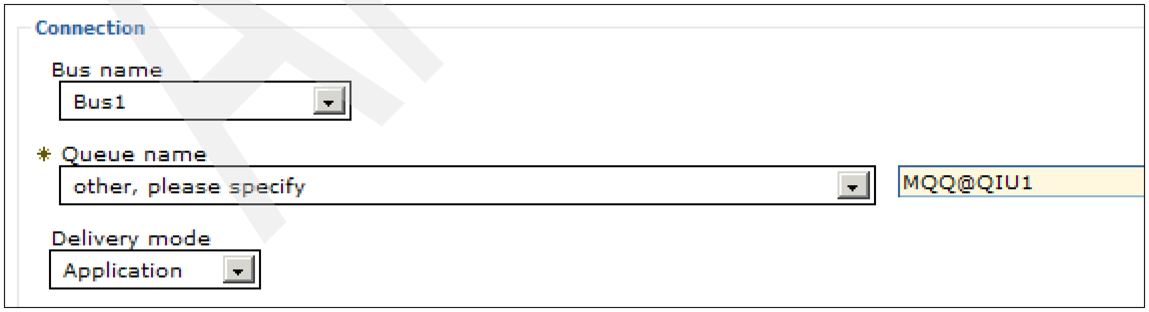
  For example, in the administrative console:

  a. Select **Resources** → **JMS** → **Queues**.

  b. Click on the JMS queue name.

  c. In the Connection area, specify the bus name where the MQ link is defined.

  d. Select **other, please specify** for the Queue name, and enter the queue name as in Figure 22-3.


Connection

Bus name: Bus1

* Queue name: other, please specify — MQQ@QIU1

Delivery mode: Application

Figure 22-3 Specifying the target queue with queue manager

22.6.6 Validate the solution

If you made changes to the WebSphere MQ link definition or JNDI resources, restart your application server configuration.

Send messages and ensure they arrive on the target WebSphere MQ destination.

22.7 Encountering a message flow problem from WebSphere MQ to the service integration bus

This topic examines some of the issues that can affect message flow from WebSphere MQ to service integration bus destinations via the WebSphere MQ link.

22.7.1 Verify integrity

Work with the WebSphere MQ administrator to:

- Determine how far through the WebSphere MQ system the message has progressed.
- Check the runtime status of the WebSphere MQ sender channel.
- Determine if the messages counter is incremented when messages are sent.

22.7.2 Collect diagnostics

Collect the following information:

- In the application server SystemOut log files, determine:
  - The server hosting the messaging engine configured with the WebSphere MQ link.
  - The server hosting the application which is attempting to receive messages.
  - The servers hosting messaging engines involved in routing the message.

  For information about collecting the SystemOut log, see 10.3, “Collect diagnostics” on page 339.

- Examine the exception destination for the messaging engines involved in your messaging flow and hosting the WebSphere MQ link. For more information about exception destinations, see *Exception destinations* at:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjo0004_.html`
- Locate the WebSphere MQ queue manager error logs. Work with your MQ administrator to collect the appropriate logs for analysis.
- Examine the dead letter queue on the appropriate queue managers on the WebSphere MQ system.

22.7.3 Analyze diagnostics

To analyze diagnostics:

- If the WebSphere MQ sender channel messages counter is *not incrementing*, the messages are not being transmitted across the channel. Examine the WebSphere MQ queue manager errors logs and dead letter queue to determine the location of the message and the reason it cannot be sent.

  If you see message MQRC_UNKNOWN_REMOTE_Q_MGR, see 22.7.4, "Target queue manager is not known" on page 517.
- If the WebSphere MQ sender channel messages counter is *incrementing*, messages are being sent across the MQ link but, for some reason, they are *stuck* or *lost* in the service integration bus.

  Examine the application server logs and the exception destinations for any applicable messaging engines to determine the location of the message and the reason the message has not been delivered.

  Common problems include:

  – The target destination does not exist so the message has been routed to the exception destination.

  – The target destination is full, and the message has been routed to the exception destination.

  – The target destination is on a messaging engine that cannot be contacted, and the message is held on a remote queue point until the message can be delivered.

  – The target destination specifies a bus that does not exist and the message has been routed to the exception destination.

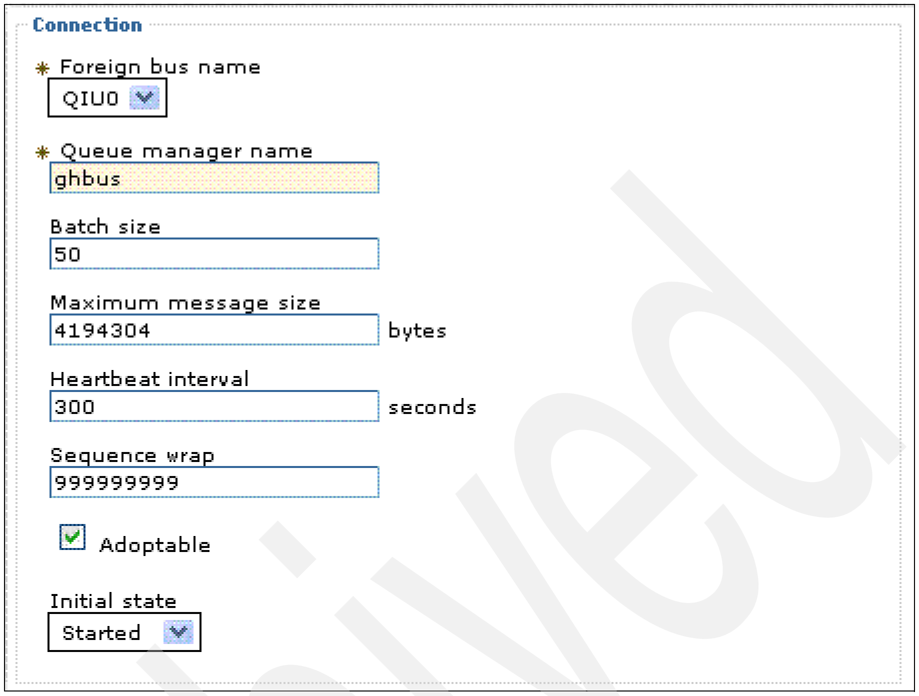
If you are producing the messages using a messaging engine that is not hosting the MQ link, you could be experiencing a problem related to inter-messaging engine communication. See Chapter 16, “Messaging in a multiple messaging engine environment” on page 429.

22.7.4 Target queue manager is not known

A common problem, particularly when using request-reply, is that the target queue manager is not known by the local queue manager. This problem, characterized by message MQRC_UNKNOWN_REMOTE_Q_MGR, might be due to a WebSphere MQ link configuration error.

The **Queue manager name** attribute (shown in Figure 22-4 on page 518) on the WebSphere MQ link represents the name of the queue manager that WebSphere Application Server represents. The WebSphere Application Server must specify the same value on WebSphere MQ as the remote queue manager, using the WebSphere administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Messaging engines.** Click the messaging engine name to open the details page.
3. Select **WebSphere MQ Links**. Click on the name of the WebSphere MQ link.



Connection

* Foreign bus name: QIU0

* Queue manager name: ghbus

Batch size: 50

Maximum message size: 4194304 bytes

Heartbeat interval: 300 seconds

Sequence wrap: 999999999

✓ Adoptable

Initial state: Started

Figure 22-4 Defining the queue manager name for the foreign bus

22.7.5 Validate the solution

If you have made changes to the WebSphere MQ link definition or JNDI resources, you must restart your application server configuration so that they take effect. To confirm the solution to the message flow problem, resend additional messages and ensure they arrive on the target service integration bus destination.

22.8 Messages flow across the WebSphere link but appear corrupt

This topic examines reasons that messages arrive at the intended destination but appear to have missing information or incorrect data in them.

22.8.1 Collect diagnostics

Collect the header and payload information of the message that appears incorrect.

The consuming application usually has an error when processing the header information of the message.

The approach for analyzing the message depends on the direction of transport:

- WebSphere MQ to WebSphere Application Server

  Look at the message on the exception destination. Alternatively, modify the consuming application to output the content of the message to a log, for example. There is a small chance the message might be rolled-back to a queue point.

- WebSphere Application Server to WebSphere MQ

  You need to work with the WebSphere MQ administrator to recover the content of the message.

22.8.2 Analyze diagnostics

Examine the message headers and payload to determine if the message is incorrect.

If the message is missing JMS specific headers, such as JMSCorrelationID or custom defined user headers, see 22.8.3, "MQRFH not included" on page 519.

22.8.3 MQRFH not included

By default, the MQRFH is not included when sending messages from a service integration bus to WebSphere across a WebSphere MQ link. As a result, any JMS specific headers such as JMSCorrelationID and custom defined user headers are not included in the message.

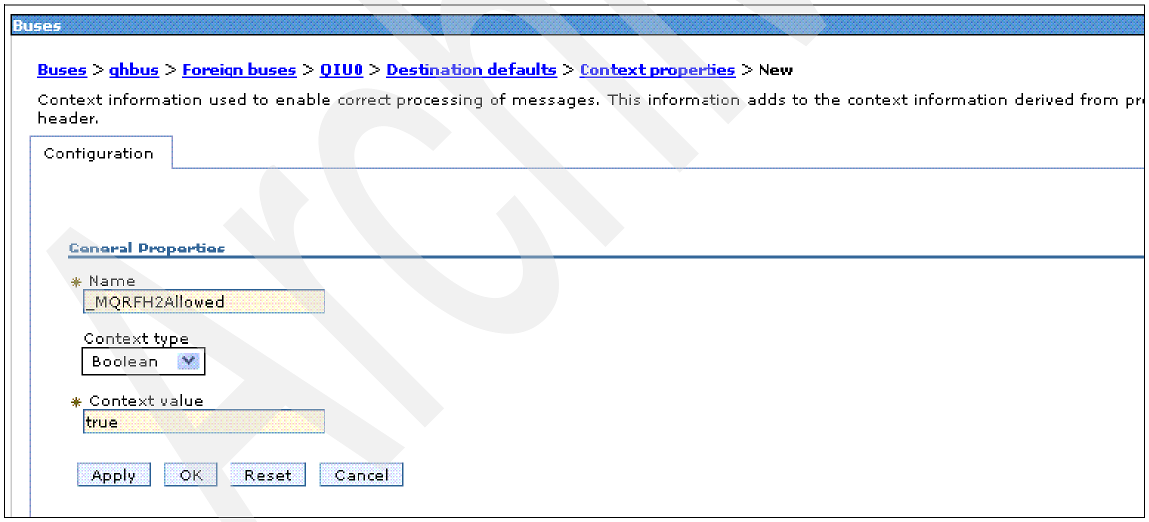
Note: This is not the case when messages flow into the service integration bus from WebSphere MQ, because all messages are converted into JMS messages.

To include the JMS headers on outbound messages, you must set the _MQRFH2Allowed context property. You can set it for individual alias destinations, foreign destinations, or on the WebSphere MQ link destination default so that all messages sent across the WebSphere MQ link will include the JMS headers. You must set this property with a Boolean value of `true`.

Using the administrative console:

1. Select **Service integration** → **Buses**. Click the bus name to open the details page.
2. Click **Foreign buses**. Click the foreign bus name to open the details page.
3. Click **Destination defaults**.
4. Click **Context properties**.
5. Click **New**:
   – Enter `_MQRFH2Allowed` for the Name.
   – Select `Boolean` for the Context type.
   – Enter `true` for the Context value.
6. Click **OK**.

Figure 22-5 shows the panel you use to set the property.



Buses > qhbus > Foreign buses > QIU0 > Destination defaults > Context properties > New

Context information used to enable correct processing of messages. This information adds to the context information derived from pr
header.

* Name: _MQRFH2Allowed

Context type: Boolean

* Context value: true

Figure 22-5 Enable MQRFH headers

22.9 Validate the solution

If you made changes to the WebSphere MQ link definition or to JNDI resources, restart the application server so that they take effect.

Send messages and ensure that the expected message properties are now correctly propagated.

JMS application with WebSphere MQ problem determination

This chapter investigates some of the common problems encountered when an application attempts to do one of the following:

- Create a connection to WebSphere MQ using the WebSphere MQ messaging provider.
- Produce (or send) a message to a WebSphere MQ destination within a JMS application.
- Consume a message from a WebSphere MQ destination.

23.1 Identify symptoms

The symptom that a JMS application is *unable to connect to WebSphere MQ* is that the application receives a `JMSException` when it attempts the connection. If this symptom fits your problem, see 23.3, “JMS application is unable to create a connection” on page 525.

Symptoms that an application is *unable to produce messages* to WebSphere MQ include:

- A JMS application receives a `JMSException` while attempting to create a producer for a given WebSphere MQ destination.
- A JMS application receives a `JMSException` while attempting to send a message.
- A JMS application appears to have sent the message successfully, but the message never arrives on the expected WebSphere MQ destination.

If these symptoms fit your problem, see 23.4, “JMS application is unable to produce messages” on page 526.

The symptom that an application is *unable to consume messages* from WebSphere MQ is that the application receives a `JMSException` when attempting to create a consumer on a given WebSphere MQ destination. If this symptom fits your problem, see 23.5, “JMS application is unable to consume messages” on page 527.

23.2 Analyze the application log

Examine the application server logs for JMS exceptions. If the root cause is a WebSphere MQ problem, look for a WebSphere MQ reason code should be included in the linked exception information.

To determine the cause of the problem , you need to identify the WebSphere MQ reason code. For example, in the section of stack trace shown below, the WebSphere MQ reason code is `2059`.

```
:Caused by: com.ibm.mqservices.MQInternalException: MQJE001: An
MQException occurred: Completion Code 2, Reason 2059
MQJE011: Socket connection attempt refused
:
```

Tip: Use the **`mqrc`** ***`reason_code`*** command on a WebSphere MQ system to get a brief description of the given reason code.

The following sections discuss some of the more common problems you might find.

23.3 JMS application is unable to create a connection

The following are some of the more common WebSphere MQ reason codes you might experience:

- Reason code `2035`

  The user ID being provided on the connection attempt is not authorized on the WebSphere MQ system. To resolve this issue:

  – Provide an authentication alias to be used by the connection that is an authorized user on the WebSphere MQ System.

  – Provide a valid user ID/password on the client application's call to the `createConnection()` method.

- Reason code `2058`

  The queue manager name provided on the connection attempt does not match the queue manager to which it has been connected.

  Two reasons this error can occur are:

  – Although the connection factory is configured with the correct host, port, and queue manager name, the connection method has been set to **`bindings`**. This is the default when defining an WebSphere MQ connection factory. The connection attempt is made locally.

    If trying to connect to a remote system, ensure the connection method is set to **`client`**.

  – The connection has successfully been made, but the queue manager name provided does not match the queue manager name with which the WebSphere MQ listener is associated.

    Ensure the queue manager name in the connection factory is correct, including the correct case.

- Reason code `2059`

  The queue manager to which you are attempting to connect is not available.

  Examine the WebSphere MQ queue manager error log for an indication of the root cause. If the reason for the failure is not apparent, check the following:

  – If the listener is running but the queue manager itself is not started, ensure the queue manager is running.

  - If the connection is trying to use a client connection but the listener is not running or the port provided on the connection factory is incorrect, ensure that the listener is running and the port is correct.

23.4 JMS application is unable to produce messages

The following are some of the more common WebSphere MQ reason codes you might experience:

- Failures that occur when creating a producer for the given WebSphere MQ destination:
  - Reason code `2035`

    The user ID is not authorized to perform this action.

    Ensure the correct user id is being used and that the correct security settings are in place for the WebSphere MQ queue manager.
  - Reason code `2051`

    The destination you are attempting to send to is put disabled.

    Investigate why the queue might be disabled, and modify the configuration appropriately.
  - Reason code `2085`

    The destination to which you are attempting to connect does not exist.

    The first line of the exception contains a message similar to this:

    ```
    MQJMS2008: failed to open MQ queue Queue_Name
    ```

    Ensure that the queue name is correct and exists within the target WebSphere MQ queue manager.
  - Reason code `2087`

    The destination to which you are attempting to send is located on a remote queue manager and has been specified in the JMS queue definition; however, the local queue manager is not configured to route messages to the remote queue manager.

    Work with the WebSphere MQ administrator to ensure that the configuration is set up to route messages between WebSphere MQ queue managers.
- A failure that occurs when sending a message produces reason code `2053`.

    The destination to which you are attempting to send already contains the maximum number of messages.

First, investigate why the queue is full and then identify appropriate actions to resolve the problem. For example, you might need to increase the maximum depth of the queue.

If the application appears to have sent the message but it does not reach the target destination, the following are possible causes:

- Message on dead letter queue.

  Check the WebSphere MQ queue manager's dead letter queue (if one is defined) to see if the message has been moved there. If this is the case, then examine the dead letter header for more information to explain why the message was moved.

- Target destination cannot be reached.

  If the target destination was hosted on a remote queue manager, ensure that the channels between queue managers are running.

23.5 JMS application is unable to consume messages

The following are some of the more common WebSphere MQ reason codes you might experience:

- Reason code `2016`

  The destination from which you are attempting to receive is inhibited.

  Investigate why the queue might be inhibited and modify the configuration appropriately.

- Reason code `2035`

  The user ID being used is not authorized to perform this action.

  Ensure the correct user ID is being used and that the correct security settings are in place for the WebSphere MQ queue manager.

- Reason code `2085`

  The destination to which you are attempting to connect does not exist. In such a case, the first line of the exception contains a message similar this:

  ```
  MQJMS2008: failed to open MQ queue Queue_Name
  ```

  Ensure that the queue name is correct and exists within the target WebSphere MQ queue manager.

23.6 Validate the solution

If you made changes to the WebSphere Application Server JMS or JNDI resources, restart the application server so that changes take effect. If changes were only made to WebSphere MQ queue manager objects, you do not normally need to restart the application server.

Run the application and ensure the problem has been resolved.

24

Foreign bus problem determination

This chapter identifies common problems that can occur when setting up a foreign bus link using the default messaging provider. These problems are related to configuration errors when defining a new foreign bus link or when configuring an application to use the link.

24.1 Introduction to foreign buses

A foreign bus is a representation of another service integration bus, providing messaging functionality from one service integration bus to another. A foreign bus link can only be used for send operations. You cannot receive messages from a remote bus across a foreign bus link.

For a detailed description of a foreign bus, see *Foreign buses* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.iseries.doc/concepts/cjj0030_.html

For examples that use foreign buses, see 24.5, “Foreign bus link usage examples” on page 533.

24.1.1 Identify symptoms

Common symptoms of a foreign bus problem include:

- A messaging engine fails to start after configuring a foreign bus link.
- A foreign bus link fails to start.
- A JMS application gets an exception while attempting to produce messages on a foreign destination.

If the messaging engine and foreign bus link both start, you might be experiencing application problems.

24.2 Verify system integrity

The first step in diagnosing the problem is to verify the status of the messaging engine and the foreign bus link.

24.2.1 Verify the status of the foreign bus link

The first step in diagnosing any foreign bus problem is to determine if the foreign bus links on the buses are active. To check the status of the foreign bus links from the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Messaging engines**.

4. Click the messaging engine name to open the details page.
5. Click **Service integration bus links** (under Additional Properties heading).

Ensure the foreign bus link is in a `Started` state (a solid green arrow icon). If the link is not started, attempt to start it by selecting the box to the left of the link and clicking **Start.**

If the link will not start, you might have a configuration problem.

24.2.2 Verify the messaging engine has started

A problem with foreign bus configuration can prevent the messaging engine from starting. Use the instructions in 10.2.3, “Determine your messaging engine status” on page 328 to ensure the messaging engine has started.

If the messaging engine is not started, attempt to start it. If the link will not start, you might have a configuration problem.

24.3 Analyze diagnostics

Problems with a foreign bus configuration are usually indicated by messages in the application server SystemOut log for the messaging engines. If the messaging engine or link to the foreign bus failed to start, you see messages that indicate the problem in this log.

If the messaging engine and foreign bus both appear to be successfully started, the problem might be related to the application.

24.3.1 Analyze SystemOut for the messaging engines

A foreign bus is a link between two messaging engines in two service integration buses. Analyze the SystemOut log for the application servers hosting both messaging engines.

Examine the SystemOut logs for these error messages:

- If you see this message, the name of the bus you defined in the foreign bus configuration does not match the name of the remote bus.

  ```
  CWSIT0057E: The inter-bus connection Foreign_Bus_Link_Name failed in
  the remote messaging engine on host Host_Name with reason:
  CWSIT0086E: Bus Bus_Name not found.
  ```

  Modify the foreign bus configuration to reference the correct remote bus name.

- If you see this message, the name of the foreign bus link you defined in this bus was not the same as the name of the foreign bus link on the other bus, or the name of the messaging engine specified in the link is incorrect.

  ```
  CWSIT0057E: The inter-bus connection Foreign_Bus_Link_Name failed in
  the remote messaging engine on host Host_Name with reason:
  CWSIT0067E: Inter-bus connection Foreign_Bus_Link_Name in bus
  Remote_Bus_Name is not available.
  ```

  Ensure that the name of the foreign bus link is the same in both buses, or correct the messaging engine name.

- If you see this message, you most likely have a configuration error related to the JMS resources used by the application.

  ```
  CWSIA0062E: Failed to create a MessageProducer for {0}
  ```

  Examine the application log for further information about the error.

- If you see this message, you have a security access error.

  ```
  CWSII0219W: The bus bus1 denied the user Bus1User access to send a
  message to the foreign bus bus2.
  ```

  See 24.5.4, “Securing foreign bus environments” on page 542 for an example of the correct way to configure the security for the foreign bus connection.

24.3.2 Analyze the application log

Examine the application log for exceptions creating the queue sender:

- If you see this message, you might have a configuration error.

  ```
  Exception caught while creating the queue sender for queue jms/q :
  javax.jms.InvalidDestinationException
  ```

  One possible cause of this error is that you have defined the JMS queue to point to the foreign destination on the local bus. It should point to the real destination on the foreign bus.

For an example of this type of configuration error, see 24.5.2, “Configuring the JMS queue to point to the correct bus” on page 536.

If you found an error message other than those listed and the information included in the messages was not sufficient to determine the root cause of the message, see Chapter 27, “The next step” on page 581.

24.4 Validate the solution

Restart the application servers and check that the messaging engines and foreign bus links start. Rerun the JMS application and ensure that messages can now be produced.

24.5 Foreign bus link usage examples

The following sections provide examples of two areas of foreign bus usage that are commonly configured incorrectly and can cause problems.

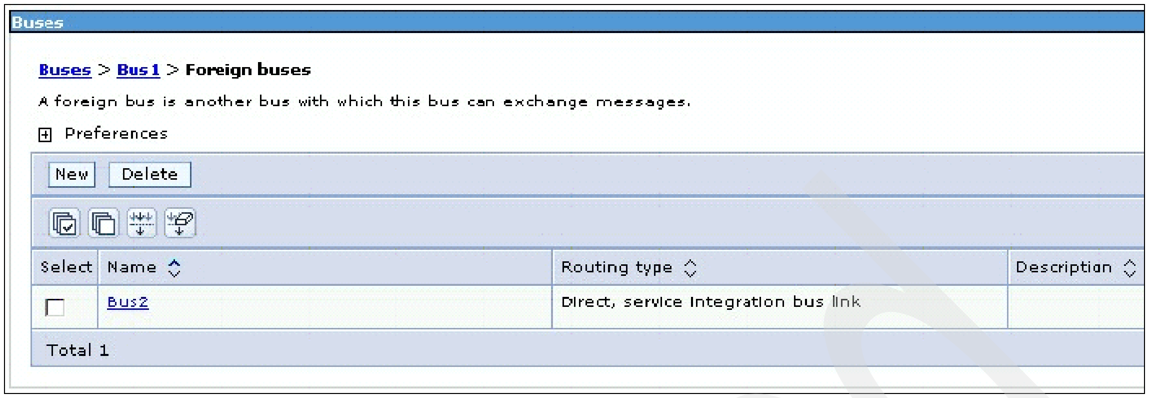
24.5.1 Establishing a foreign bus

The following example illustrates how a foreign bus configuration is established between two existing buses (Bus1 and Bus2), each with a single messaging engine, and is used as a reference throughout this topic.

An application has been deployed into an application server scope where Bus1 is defined as an existing local bus. The application needs to access resources on another bus, Bus2. To fulfill this requirement, you need to update both existing configurations to provide direct access between them. The following steps show how this might be achieved.

On Bus1:

1. Create a foreign bus definition, which has the same name as the remote bus (in this case, `Bus2`). See Figure 24-1 on page 534.



Buses

Buses > Bus1 > Foreign buses

A foreign bus is another bus with which this bus can exchange messages.

⊞ Preferences

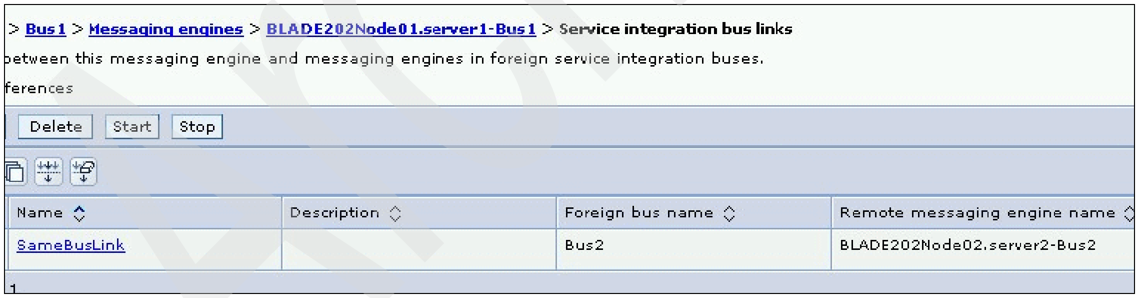
Select	Name	Routing type	Description
☐	Bus2	Direct, service integration bus link	
Total 1			

Figure 24-1   Foreign bus definition on Bus1

Note: You can also define other routing properties for the foreign bus (on both buses) for security and topic space mapping. In this example, they are left unchanged.

2. Define a foreign bus link, (for example, named `SameBusLink`), that has Bus2 as its foreign bus name and a remote messaging engine definition with the name of a messaging engine in Bus2. See Figure 24-2.

Tip: Copy the name of the messaging engine into the clipboard before creating this link.



> Bus1 > Messaging engines > BLADE202Node01.server1-Bus1 > Service integration bus links

etween this messaging engine and messaging engines in foreign service integration buses.

ferences

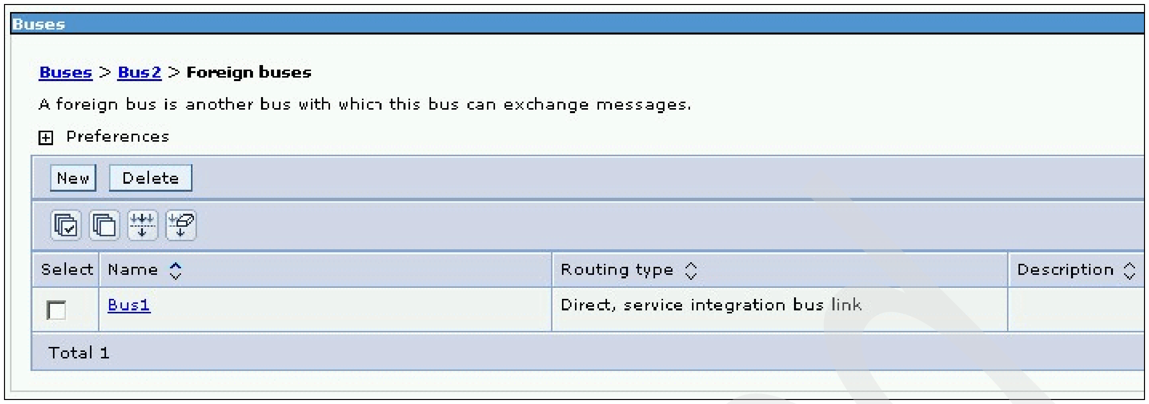
Name	Description	Foreign bus name	Remote messaging engine name
SameBusLink		Bus2	BLADE202Node02.server2-Bus2

1

Figure 24-2   Foreign bus link

On Bus2:

1. Create a foreign bus definition. Use the same name as the remote bus, in this case, `Bus1`. See Figure 24-3 on page 535.


Buses

Buses > Bus2 > Foreign buses

A foreign bus is another bus with which this bus can exchange messages.

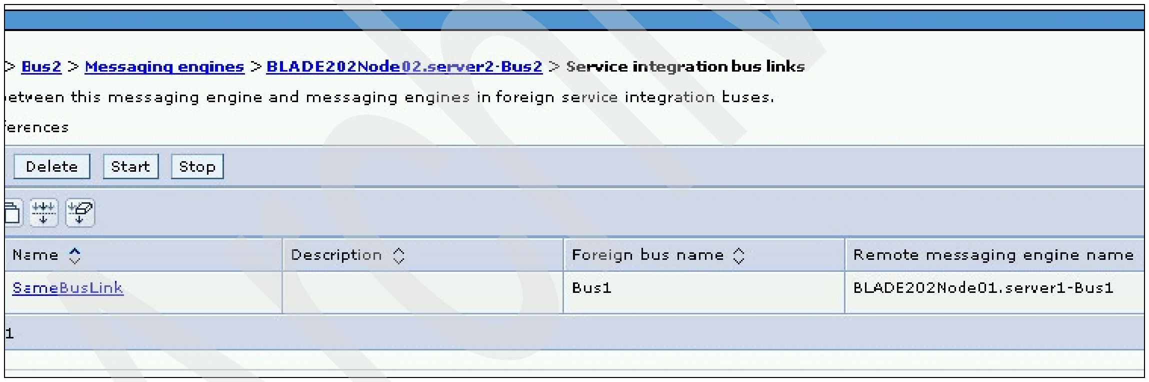
⊞ Preferences

Select	Name	Routing type	Description
☐	Bus1	Direct, service integration bus link	

Total 1

Figure 24-3   Foreign bus on Bus2

2. Define a foreign bus link with the following properties:
   – Foreign bus link name: `SameBusLink`
   – Foreign bus: `Bus1`
   – Remote messaging engine: The name of a messaging engine in Bus1. See Figure 24-4.


> Bus2 > Messaging engines > BLADE202Node02.server2-Bus2 > Service integration bus links

etween this messaging engine and messaging engines in foreign service integration buses.

erences

Name	Description	Foreign bus name	Remote messaging engine name
SameBusLink		Bus1	BLADE202Node01.server1-Bus1

1

Figure 24-4   Foreign bus link

In point-to-point messaging, you can only send messages to a remote queue; the remote queue cannot receive messages.

A foreign bus is really a link between a messaging engine in one bus and a messaging engine in another. To enable the application server to process messages across to the foreign bus, you must configure a foreign bus, and both messaging engines and both foreign bus links must be started. When the bus link is started on one bus, both sides of the link usually become active.

For a detailed description of configuring foreign buses, see *Configuring the properties of a foreign bus* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.iseries.doc/tasks/tjj0078_.html

Deleting and recreating or modifying the properties on only one side of a foreign bus link can cause problems. You should delete, and then recreate the complete foreign bus link configuration with the correct properties. Then, restart the application servers so that the foreign bus configuration changes take effect.

24.5.2 Configuring the JMS queue to point to the correct bus

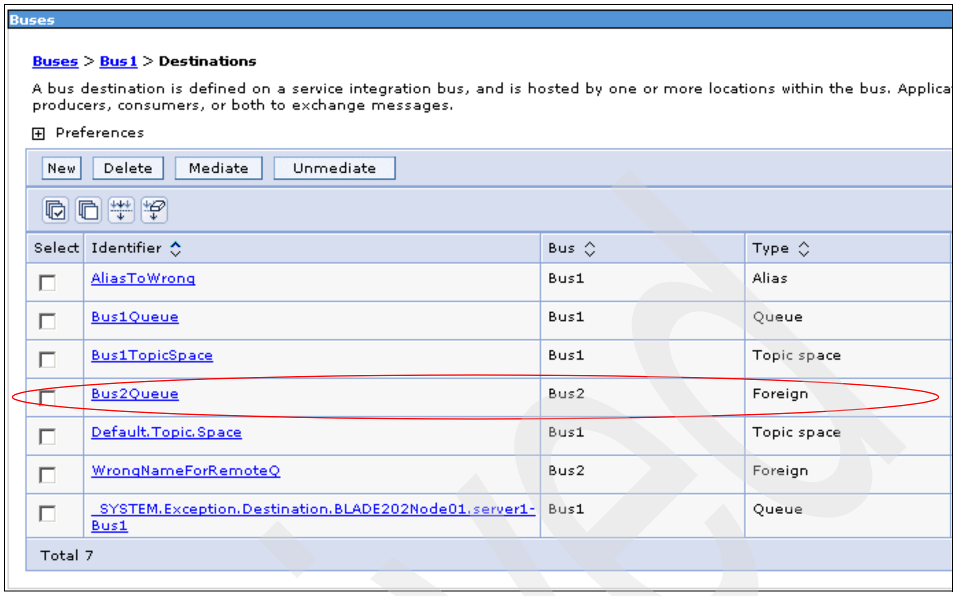
Configuration errors are the most common type of problem experienced when attempting to use a foreign destination to produce messages to a queue on a remote bus using a JMS client and JMS queue definitions.

Perhaps the most common problem is configuring the JMS queue to point to the incorrect destination.

Note: The JMS queue must point to the real destination on the remote (foreign) bus, not the foreign destination on the local bus.

For example, consider the situation in which you have an application that wants to produce messages to Bus2Queue. Bus2Queue resides on bus2 and is defined as a foreign destination to Bus1.

Figure 24-5 on page 537 shows the definition for Bus2Queue from the viewpoint of Bus1.



Buses

Buses > Bus1 > Destinations

A bus destination is defined on a service integration bus, and is hosted by one or more locations within the bus. Applica
producers, consumers, or both to exchange messages.

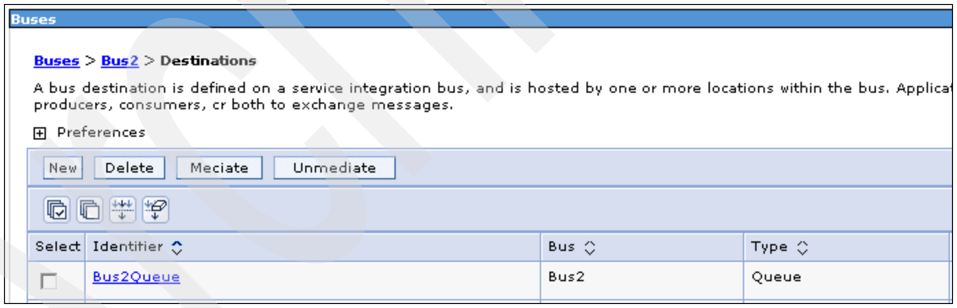
⊞ Preferences

Select	Identifier	Bus	Type
☐	AliasToWrong	Bus1	Alias
☐	Bus1Queue	Bus1	Queue
☐	Bus1TopicSpace	Bus1	Topic space
☐	Bus2Queue	Bus2	Foreign
☐	Default.Topic.Space	Bus1	Topic space
☐	WrongNameForRemoteQ	Bus2	Foreign
☐	_SYSTEM.Exception.Destination.BLADE202Node01.server1-Bus1	Bus1	Queue

Total 7

Figure 24-5 Bus2Queue defined as a foreign destination to Bus1

Figure 24-6 shows the configuration of Bus2Queue from the viewpoint of Bus2. Bus2Queue is a queue on Bus2.



Buses

Buses > Bus2 > Destinations

A bus destination is defined on a service integration bus, and is hosted by one or more locations within the bus. Applica
producers, consumers, or both to exchange messages.

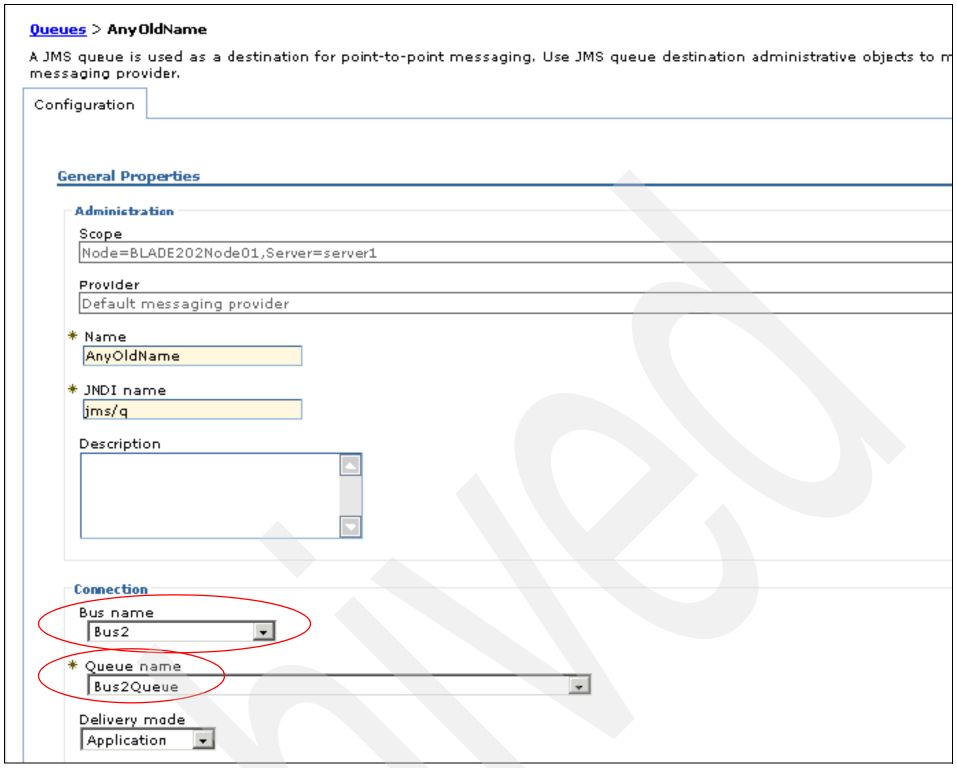
⊞ Preferences

Select	Identifier	Bus	Type
☐	Bus2Queue	Bus2	Queue

Figure 24-6 Bus2Queue defined as a topic space on Bus2

Correct configuration

Figure 24-7 on page 538 shows the correct configuration for JMS queue `jms/q` and destination Bus2Queue. The JMS queue references the real destination on the remote (foreign) bus, Bus2.


Queues > AnyOldName

A JMS queue is used as a destination for point-to-point messaging. Use JMS queue destination administrative objects to m
messaging provider.

Configuration

General Properties

Administration

Scope
Node=BLADE202Node01,Server=server1

Provider
Default messaging provider

* Name
AnyOldName

* JNDI name
jms/q

Description

Connection

Bus name
Bus2

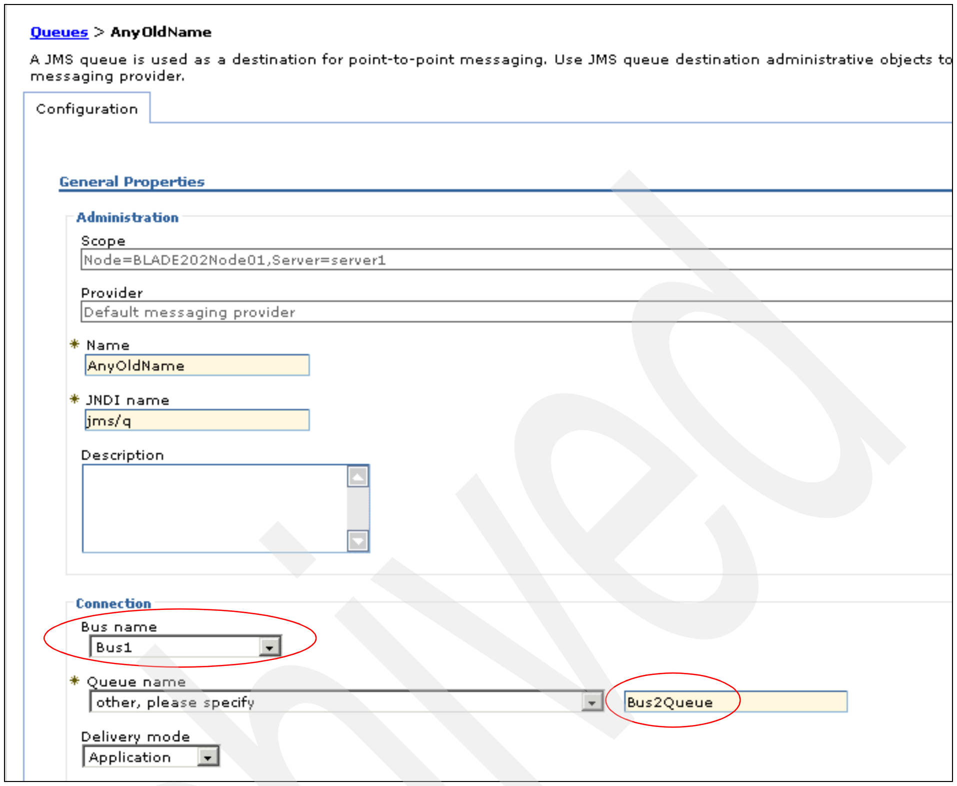
* Queue name
Bus2Queue

Delivery mode
Application

Figure 24-7 Correct configuration for a queue on a foreign destination

Incorrect configuration

Figure 24-8 on page 539 illustrates an incorrect configuration for JMS queue `jms/q` and destination Bus2Queue. The JMS queue is bound to a destination that does not exist on Bus1 because it only references a foreign destination.


Queues > AnyOldName

A JMS queue is used as a destination for point-to-point messaging. Use JMS queue destination administrative objects to
messaging provider.

Configuration

General Properties

Administration

Scope: Node=BLADE202Node01,Server=server1

Provider: Default messaging provider

* Name: AnyOldName

* JNDI name: jms/q

Description:

Connection

Bus name: Bus1

* Queue name: other, please specify — Bus2Queue

Delivery mode: Application

Figure 24-8   Incorrect: JMS queue points to the destination on the local bus

When a JMS application attempts to produce messages on this JMS queue, the following error message is written to the application server log files.

```
CWSIA0062E: Failed to create a MessageProducer for
queue://<Remote_Bus_Name+Destination_Name>?busName=Local_Bus_Name
at:
```

24.5.3 Configuring topic space mappings

Topic space mapping enables subscribers on a local topic space to receive messages that have been published to a remote topic space using the foreign bus link. The publisher and subscribers on both buses must both share the same topic name.

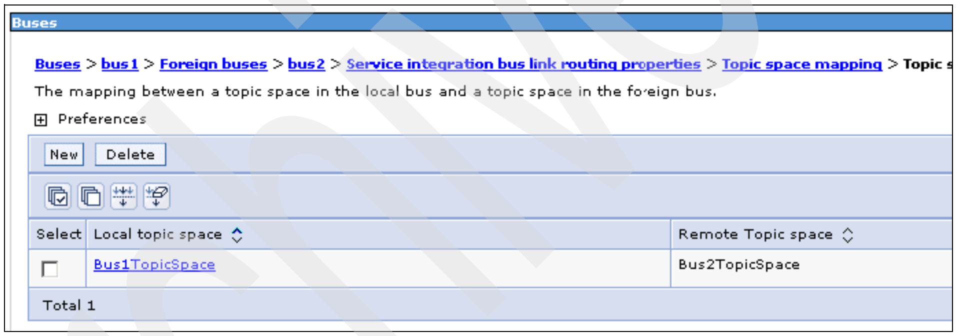
For more detailed information about topic space mappings, see *Configuring topic space mappings between service integration buses* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.doc/tasks/tjj0707_.html

In this example, assume that:

- Messages are published to a remote topic space called `Bus2TopicSpace` on bus2.
- Messages are subscribed to from a local topic space called `Bus1TopicSpace` on bus1.
- The topic name is `Fruit`.
- Messages are sent using JMS clients.

The mapping is defined in the local bus, bus1, as a routing property of the foreign bus, bus2, as shown in Figure 24-9.

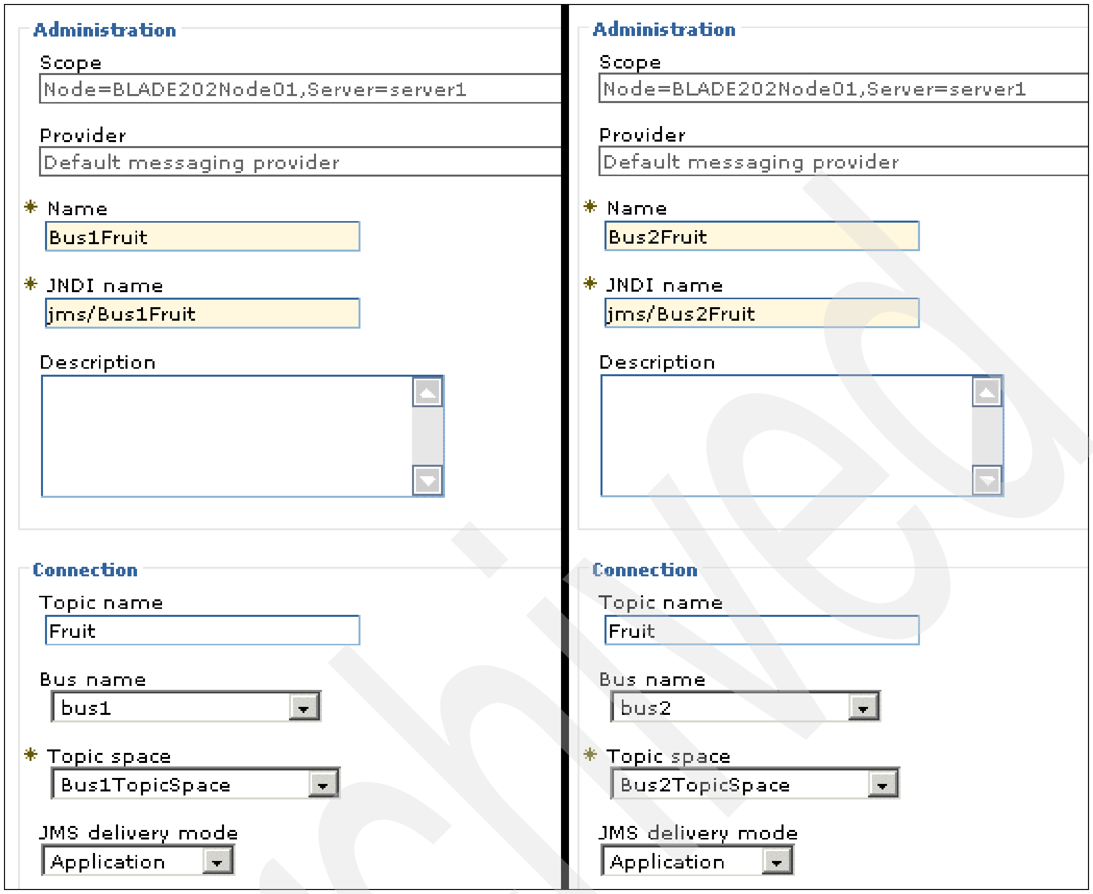

Buses
Buses > bus1 > Foreign buses > bus2 > Service integration bus link routing properties > Topic space mapping > Topic s
Preferences
New
Delete

The mapping between a topic space in the local bus and a topic space in the foreign bus.

Select	Local topic space	Remote Topic space
☐	Bus1TopicSpace	Bus2TopicSpace
Total 1		

Figure 24-9 Bus1TopicSpace mapped to Bus2TopicSpace

The JMS topic definitions in Figure 24-10 on page 541 are provided for reference. On the left, you see the definitions for Bus1Fruit; on the right, Bus2Fruit.



Administration

Scope: Node=BLADE202Node01,Server=server1

Provider: Default messaging provider

* Name: Bus1Fruit

* JNDI name: jms/Bus1Fruit

Description:

Connection

Topic name: Fruit

Bus name: bus1

* Topic space: Bus1TopicSpace

JMS delivery mode: Application

Administration

Scope: Node=BLADE202Node01,Server=server1

Provider: Default messaging provider

* Name: Bus2Fruit

* JNDI name: jms/Bus2Fruit

Description:

Connection

Topic name: Fruit

Bus name: bus2

* Topic space: Bus2TopicSpace

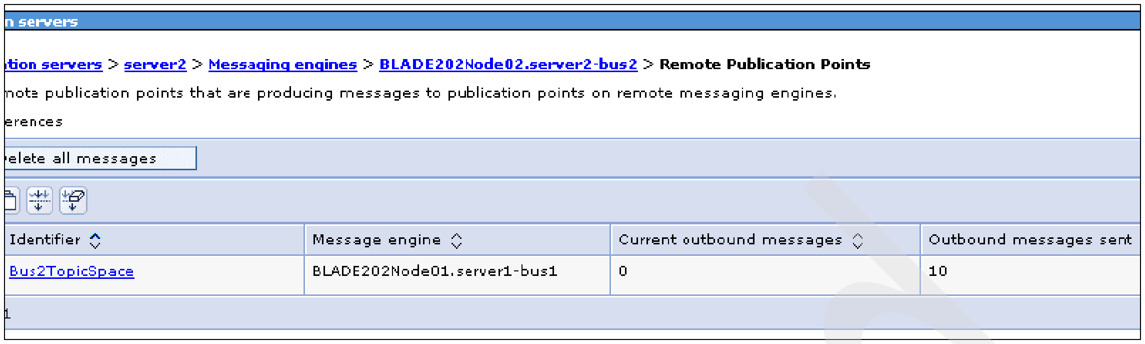
JMS delivery mode: Application

Figure 24-10 Topic definitions on bus1 and bus2

Using the WebSphere Application Server administrative console, to determine whether messages have flowed across the foreign bus link:

1. Select **Servers** → **Application servers**.
2. Click the server name hosting bus2 (for this example, server2).
3. Click **Messaging engines**.
4. Click the messaging engine name to open the details page.
5. Click the **Runtime** tab.
6. Under the Remote message points heading, click **Remote publication points**.

This panel shown in Figure 24-11 on page 542 displays.


n servers

tion servers > server2 > Messaging engines > BLADE202Node02.server2-bus2 > Remote Publication Points

mote publication points that are producing messages to publication points on remote messaging engines.

erences

elete all messages

Identifier	Message engine	Current outbound messages	Outbound messages sent
Bus2TopicSpace	BLADE202Node01.server1-bus1	0	10

1

Figure 24-11 Remote publication points for server2

The **Outbound messages sent** column reflects the number of messages that have flowed across the link.

24.5.4 Securing foreign bus environments

This topic builds on the WebSphere Application Server V6.1 Information Center task, *Securing access to a foreign bus* at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.iseries.doc/tasks/tjj0002S_.html`

The following steps provide an example of securing a foreign bus configuration using point-to-point messaging.

The local bus is referred to as Bus1 and the remote (foreign) bus is Bus2. It assumes two user IDs (one for each bus) have been created,: Bus1User and Bus2User, respectively.

To secure a foreign bus:

1. Enable administrative security, and configure a user repository. In this example, both the local bus and the remote (foreign) bus use the same repository.

   For more information about performing this task see *Administrative security* at:

   `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/csec_global.html`

2. Stop your WebSphere Application Server configuration.

3. Enable bus security for both Bus1 and Bus2 as documented in *Messaging security* at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.iseries.doc/concepts/cjr0420_.html

4. Give Bus1User the bus connector role for Bus1. Using the administrative console:

   a. Select **Service integration** → **Buses**.

   b. Click on the link under the Security heading for Bus1.

   c. Select **Users and groups in the bus connector role**.

   d. Click **New**, and add user Bus1User.

5. Use the previous steps to give Bus2User the bus connector role for Bus2.

6. Create an authentication alias for the foreign bus link; for example, FBLAlias setup for user Bus1User:

   a. Select **Service integration** → **Buses**.

   b. Click on the link under the Security heading for Bus1.

   c. Select **JAAS - J2C authentication data**.

   d. Click **New** and add the alias.

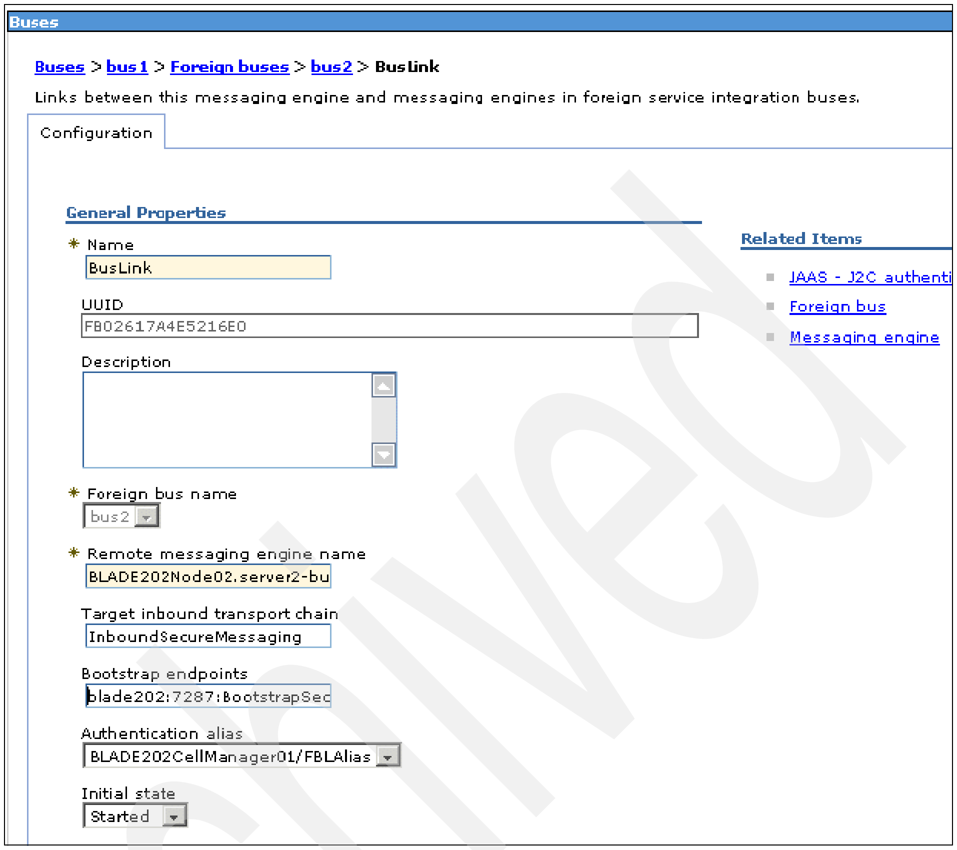
7. Update the foreign bus definition for Bus2 on Bus1:

   a. Select the authentication alias to be used to authenticate to Bus2.

   b. Add the Target inbound transport chain and Bootstrap endpoints to use the secure chain and port, as shown in Figure 24-12 on page 544.

      This procedure defines the link so it connects to Bus2 using a particular messaging engine and the secure inbound transport chain. It will bootstrap using the port and chain supplied.

      Port 7287 is the SIB_ENDPOINT_SECURE_ADDRESS port for the application server hosting the messaging engine on Bus2 (host blade202).


Buses

Buses > bus1 > Foreign buses > bus2 > Buslink

Links between this messaging engine and messaging engines in foreign service integration buses.

Configuration

General Properties

* Name
BusLink

UUID
FB02617A4E5216E0

Description

* Foreign bus name
bus2

* Remote messaging engine name
BLADE202Node02.server2-bu

Target inbound transport chain
InboundSecureMessaging

Bootstrap endpoints
blade202:7287:BootstrapSec

Authentication alias
BLADE202CellManager01/FBLAlias

Initial state
Started

Related Items

- JAAS - J2C authenti
- Foreign bus
- Messaging engine

Figure 24-12 Securing the foreign bus link on Bus1

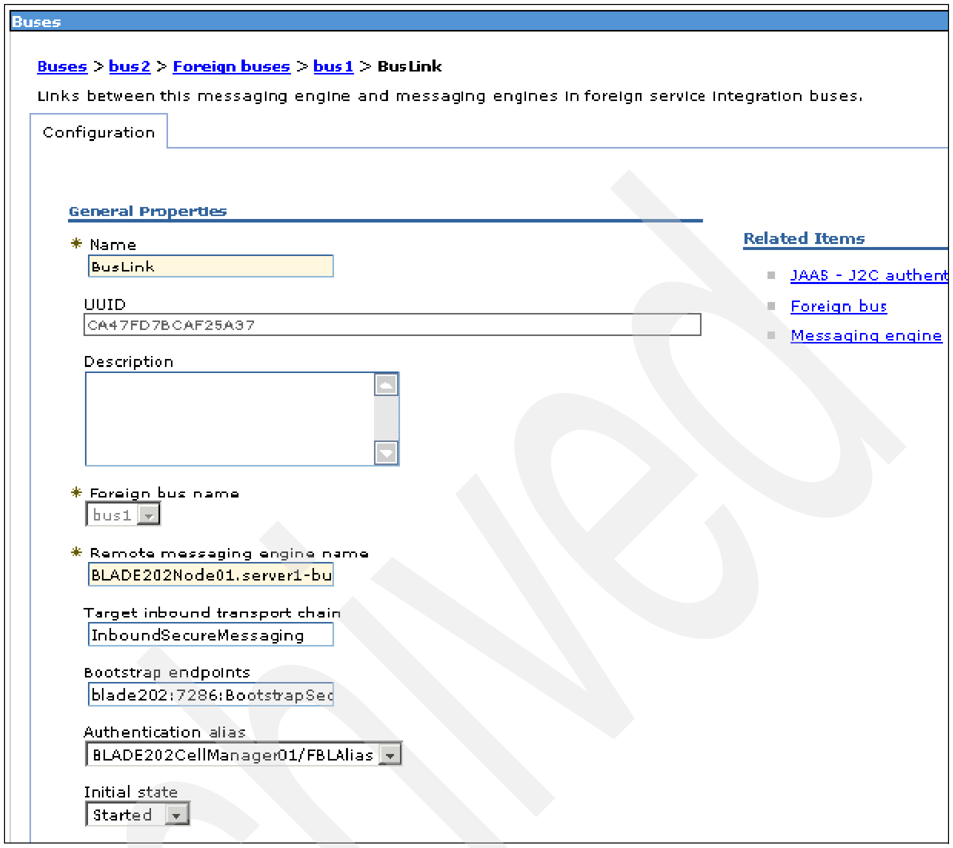
8. Update the foreign bus definition for Bus1 on Bus2:

   a. Select the authentication alias to be used to authenticate to Bus1.

   b. Add the target inbound transport chain and bootstrap endpoints to use the secure chain and port, as shown in Figure 24-13 on page 545.

      This will define the link so it connects to Bus1 using a particular messaging engine and the secure inbound transport chain. It will bootstrap using the port and chain supplied.

      Port 7286 is the SIB_ENDPOINT_SECURE_ADDRESS port for the application server hosting the messaging engine on Bus1 (host blade202).


Buses

Buses > bus2 > Foreign buses > bus1 > BusLink

Links between this messaging engine and messaging engines in foreign service integration buses.

Configuration

General Properties

* Name
BusLink

UUID
CA47FD7BCAF25A37

Description

* Foreign bus name
bus1

* Remote messaging engine name
BLADE202Node01.server1-bu

Target inbound transport chain
InboundSecureMessaging

Bootstrap endpoints
blade202:7286:BootstrapSec

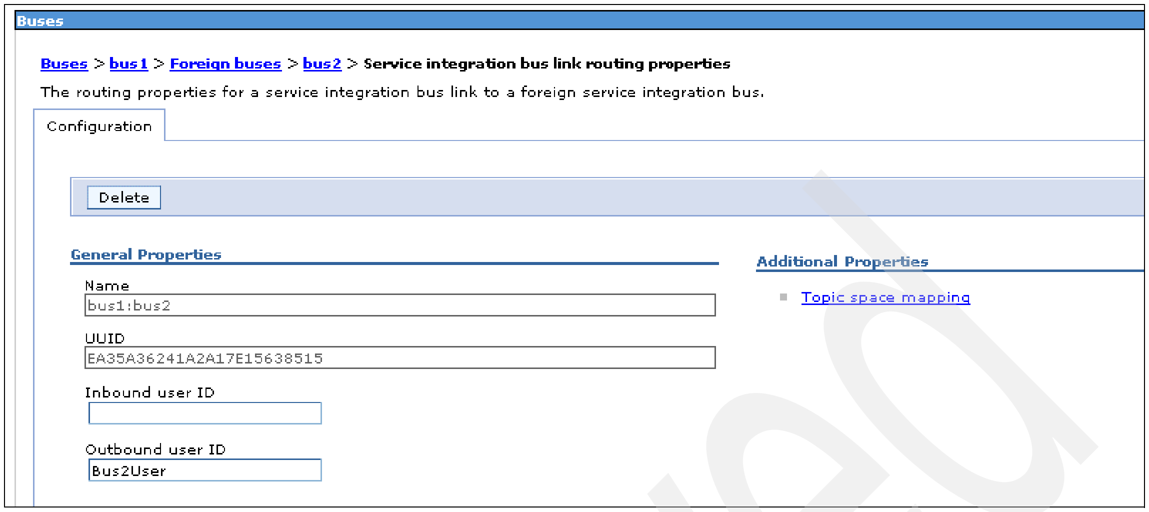
Authentication alias
BLADE202CellManager01/FBLAlias

Initial state
Started

Related Items
- JAAS - J2C authent
- Foreign bus
- Messaging engine

Figure 24-13 Securing the foreign bus link on Bus2

9. For Bus1, set the foreign bus, Bus2, routing properties to use the outbound user ID of Bus2User:
   a. Select **Service integration** → **Buses**.
   b. Click on the bus name, **Bus1**.
   c. Select **Foreign buses**.
   d. Click **Bus2**.
   e. Select **Service integration bus link routing properties**, and set the properties as shown in Figure 24-14 on page 546.



Buses > bus1 > Foreign buses > bus2 > Service integration bus link routing properties

The routing properties for a service integration bus link to a foreign service integration bus.

Name: bus1:bus2

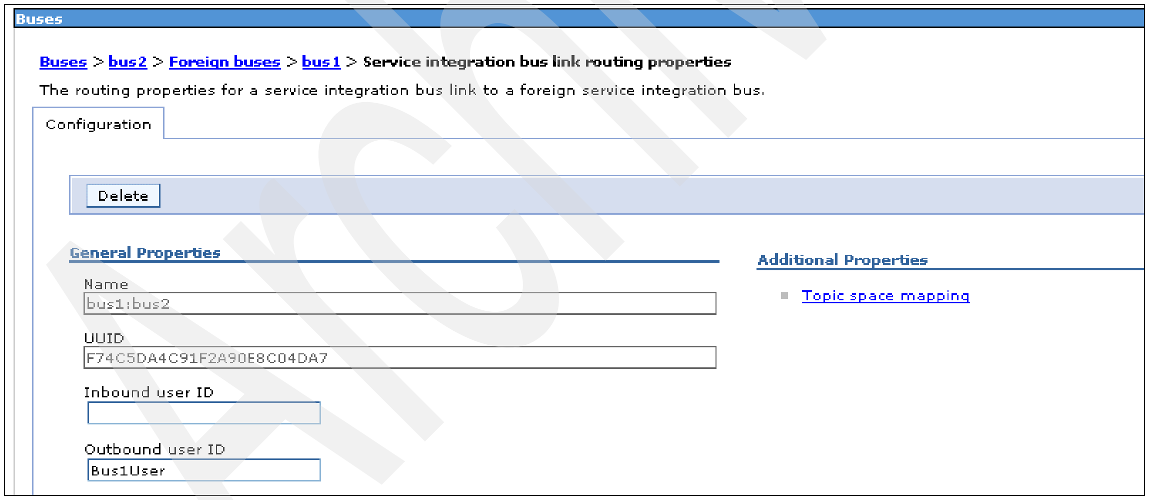
UUID: EA35A36241A2A17E15638515

Inbound user ID:

Outbound user ID: Bus2User

Figure 24-14   Service integration bus link routing properties

10. Using similar steps for Bus2, set the foreign bus, Bus1, routing properties to use the outbound user ID of Bus1User, as shown in Figure 24-15.



Buses > bus2 > Foreign buses > bus1 > Service integration bus link routing properties

The routing properties for a service integration bus link to a foreign service integration bus.

Name: bus1:bus2

UUID: F74C5DA4C91F2A90E8C04DA7

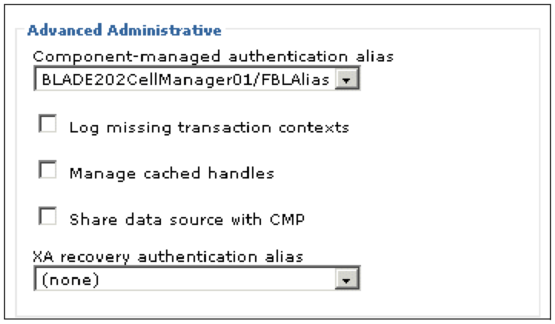
Inbound user ID:

Outbound user ID: Bus1User

Figure 24-15   Service integration bus link routing properties

11. Set the JMS queue connection factories to use the authentication alias.

For example, see Figure 24-16 on page 547.


Advanced Administrative

Component-managed authentication alias: BLADE202CellManager01/FBLAlias

☐ Log missing transaction contexts

☐ Manage cached handles

☐ Share data source with CMP

XA recovery authentication alias: (none)

Figure 24-16 Queue connection factories → Default messaging provider

12. For applications running in a WebSphere Application Server client container, you might need to use the **clientconfig.bat** tool to add an authentication alias so the application might access the secured resources.

    For more information, see *Starting the Application Client Resource Configuration Tool and opening an EAR file* at:

    http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/ucli_tstartacrct.html

13. Restart the application server, and check that the foreign bus links start.

At this point, if you attempted to put messages to the remote queue on Bus2 you would see an error message similar to the following, because you must also define the SIB access role types for the resources you are trying to use.

```
CWSII0219W: The bus bus1 denied the user Bus1User access to send a
message to the foreign bus bus2.
```

In this example, the application sends messages to the remote queue Bus2Queue on a remote (foreign) bus, Bus2. Depending on the configuration you choose, you can use one of two ways to achieve this behavior:

- If there are no foreign or alias destinations configured for the destination on Bus2, Bus1User can send messages using the foreign bus link. Because the outbound user has been specified, you must give Bus1User authority to put (or send) messages to Bus2Queue. For this example, grant privileges using the wsadmin interface:

  (Using Jacl) `$AdminTask addUserToForeignBusRole {-bus bus1 -foreignBus bus2 -role sender -user Bus1User}`

  (Using Jython) `AdminTask.addUserToForeignBusRole('[-bus bus1 -foreignBus bus2 -role Sender -user Bus1User]')`

- If a foreign or alias destination has been defined, you must give Bus1User authority to send messages to the remote destination using this foreign destination. For this example, using the wsadmin interface to grant the privileges:

  (Using Jacl) `$AdminTask addUserToDestinationRole {-bus bus1 -type ForeignDestination -foreignBus bus2 -destination Bus2Queue -role sender -user Bus1User}`

  (Using Jython) `AdminTask.addUserToDestinationRole('[-bus bus1 -type ForeignDestination -foreignBus bus2 -destination Bus2Queue -role sender -user Bus1User]')`

After restarting all processes, the user is enabled to send messages across the foreign bus link with security enabled.

Where SSL chains are used, it is necessary to exchange SSL certificates. For detailed information about how to perform this task, see *Controlling which foreign buses can link to your bus* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/tasks/tjr0420_.html

25

Default messaging provider security

This chapter discusses possible configuration problems that can occur when you configure security for the default messaging provider.

25.1 Introduction

The goal of messaging security is to allow trusted clients to perform messaging operations, and to prevent non-trusted clients from performing similar messaging operations. Various problems can occur when attempting to use messaging that are related to the security policy preventing access. These problems generally manifest themselves in one of two ways:

- An application cannot connect to the bus.
- An application, which is connected to the bus, cannot send or receive messages.

For more information about messaging security, see these additional resources:

- *Service integration bus security* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjr9999_.html

- *WebSphere MQ security* at:

  http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/fa12730_.htm

25.2 Identify symptoms

The potential problem areas related to messaging security might be classified as:

- Authentication problems connecting to the bus

  When connecting to a bus the first thing that occurs when security is enabled is to identify the client. This identity is then used for subsequent authorization checks. The identification process, commonly known as *authentication*, involves taking an identity (or user name) and a claim (or password) and checking that the claim matches the identity. If they do, the identity is confirmed. Authentication problems are typically caused by an invalid identity claim (for example, the password has expired or does not match the password stored in the user repository).

  For information about configuring the cell to contact the user repository, see the WebSphere Application Server Information Center topic, *Authenticating users* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_authusers.html

- Authorization problems connecting to the bus

  Authorization is the process of determining if an identified user is allowed to access a resource. An authorization failure does not necessarily indicate a problem (for example, when a user is not supposed to have access).

  When the application fails authorization for the bus, it is prevented from accessing the messaging resources.

- Authorization problems accessing a destination

  When the application fails authorization for a destination resource, the requesting application is not allowed access to the destination.

- SSL problems connecting to the bus

  In WebSphere Application Server V6.1, a change was made to the use of certificates for the application server . Prior to V6.1, each application server came with the same certificate, by default.

  When a profile is created in V6.1, a set of keys, specific to that server, is created. When a node is federated, the federation process ensures that trust is established within the cell. However, trust is not automatically established for clients or for cross cell communications.

25.3 Analyze diagnostics

Security problems that occur with the default messaging provider usually produce a message in the SystemOut log for the application server that is running the messaging engine, and an exception in the application log.

Look at both the top level `JMSSecurityException` or `JMSException` as well as any linked exceptions. The linked exceptions provide additional documentation and debug aids. They are subject to change without notice so you should not rely on them for program flow.

The following list contains specific errors addressed by this chapter. If you find an error message other than those listed and the information included in the message is not enough to determine the root cause of the message, see Chapter 27, “The next step” on page 581:

- This message in the SystemOut for the messaging engine indicates an authentication failure:

  CWSII0205W: `The bus` *`Bus_name`* `could not authenticate the user` *`User`*

The client JMS application receives a `JMSSecurityException` with this message:

```
CWSIA0004E: The authentication for the supplied user name User and
the associated password was not successful
```

If you find these indicators, see 25.4, "Authentication problems connecting to the bus" on page 554.

- Another example of an authentication failure is this message:

  ```
  CWSII0212W: The bus Bus_name denied an anonymous user access to the
  bus.
  ```

  This message indicates that a client has attempted to connect without providing any credentials.

  If you find this message, see 25.4, "Authentication problems connecting to the bus" on page 554.

- This message in the SystemOut for the messaging engine indicates an authorization failure for the bus:

  ```
  CWSII0211W: The bus Bus_name denied the user User access to the bus.
  ```

  The client JMS application receives a `JMSSecurityException` with this message:

  ```
  CWSIA0006E: The authorization for the supplied user name was not
  successful.
  ```

  If you find these indicators, see 25.5, "Authorization problems connecting to the bus" on page 559.

- Authorization problems accessing a destination produce messages in the range of CWSII0213-CWSII0240 and CWSII0242-CWSII0259, inclusive. The messages you see depend on what the client application was attempting to do:

  – A failure to *send* a message results in these messages:

    In SystemOut for the messaging engine server:

    ```
    CWSII0213W: The bus Bus_Name denied the user User access to send
    messages to the destination Destination_Name.
    ```

    The client JMS application receives a `JMSSecurityException`, which contains this message:

    ```
    CWSIA0069E: The user does not have authorization to carry out this
    operation. See the linked exception for details.
    ```

    The linked exception contains:

    ```
    CWSIK0018E: Send access to destination Destination_Name was denied
    for user with subject User_Name.
    ```

– A failure to *receive* a message results in these messages:

  In SystemOut for the messaging engine server:

  ```
  CWSII0214W: The bus Bus_Name denied the user User_Name access to
  receive messages from the destination Destination_Name.
  ```

  The client JMS application receives a `JMSSecurityException` with this message:

  ```
  CWSIA0090E: The user does not have authorization to carry out this
  operation. See the linked exception for details.
  ```

  The linked exception contains:

  ```
  CWSIP0309E: Receive access from destination Destination_Name was
  denied for user with subject User_Name.
  ```

If you see these indications, see 25.6, "Authorization problems accessing a destination" on page 562.

► If the failure occurs in an application server, this message is written to the application sever log files:

```
CWPKI0022E: SSL HANDSHAKE FAILURE:  A signer with SubjectDN
"CN=pdServer.itso.ibm.com, O=IBM, C=US" was sent from target
host:port "pdServer.itso.ibm.com:7287".  The signer may need to be
added to local trust store
"/opt/HDR61/profiles/jestred_na_1/config/cells/pdCell/trust.p12"
located in SSL configuration alias "NodeDefaultSSLSettings" loaded
from SSL configuration file "security.xml".  The extended error
message from the SSL handshake exception is: "No trusted certificate
found".
```

In the client, a `JMSException` is seen. More diagnostic information is contained in the linked exceptions. If the problem is a trust store issue, you see an `SSLHandshakeException` with the message:

```
No trusted certificate found
```

Problems related to the use of SSL are outside the scope of this problem determination guide. However, problems arising from trust store are common, and there are a number of useful WebSphere Information Center documents.

For information about configuring trust stores, see:

– *Retrieving signers form a remote SSL port* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_sslretrievesignersport.html

– *Adding a signer certificate to a keystore* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_ssladdsignercert.html

  - *Exchanging signer Certificates* at:

    `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_sslexchangesigncerts.html`

  For information about the **`retrieveSigners`** command used by the client, see:

  Retrieving signers using retrieveSigners utility at the client at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_sslretrievesignclient.html`

25.4 Authentication problems connecting to the bus

The two primary reasons for an authentication failure to the bus are:

- The user ID does not exist.
- The password specified is not correct.

The resolution to the problem depends on the following factors:

- Is the JMS application running in an application server or a client?
- Is the JMS application making use of container or application provided authentication?

25.4.1 Solution for an application in an application server

To resolve this problem depends on whether container or application authentication is being used. To determine which type of authentication is being used, you need to look at the application configuration in the administration console:

1. Select **Applications.**
2. Select **Manage Applications.**
3. Click on the application name that is experiencing the failure.

   Under the Resources heading, look for a **Resource references** link. If this link is not available, then your application does not have any resource references defined, which means one of two things:

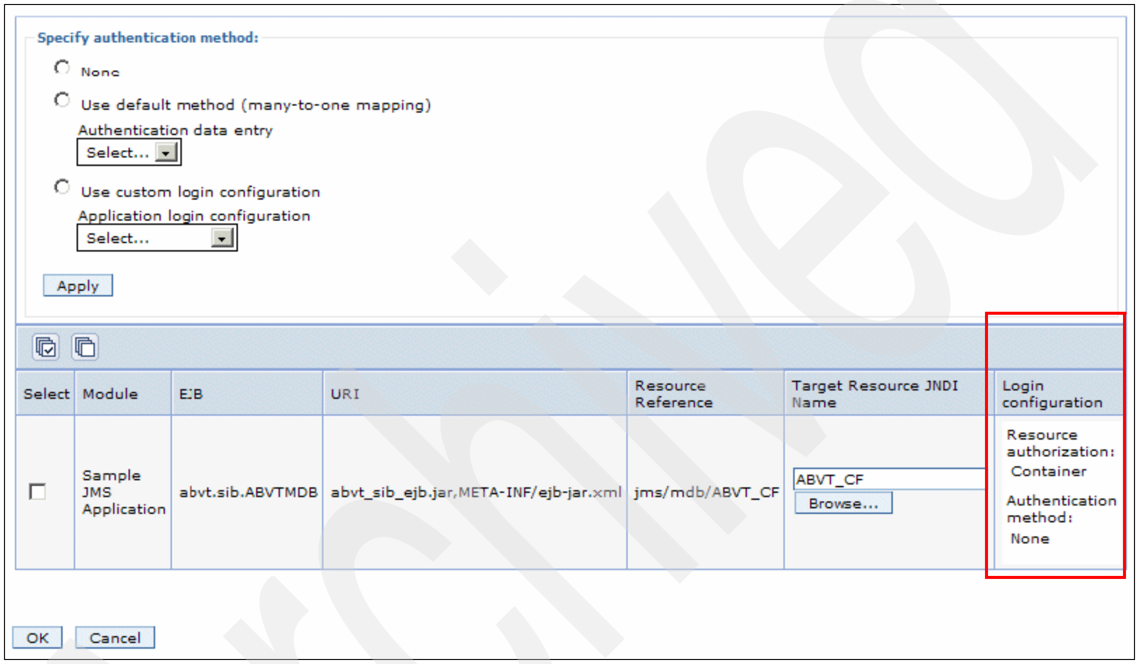
   - The application selected does not use JMS.

     If the application does not use JMS, identify the application that needs correction and start over.

   - The application selected does not use resource references.

If it does use JMS, then it is doing a direct JNDI lookup. In this case see "Resolving the problem when using application authentication" on page 556.

Click the **Resource references** link and you see a page similar to that shown in Figure 25-1. The login configuration for each resource reference is shown in the table in a red box. In this example, the authentication mode used is `Container`. The word `Container` is replaced with `Application` if application-based authentication is to be used.



Specify authentication method:

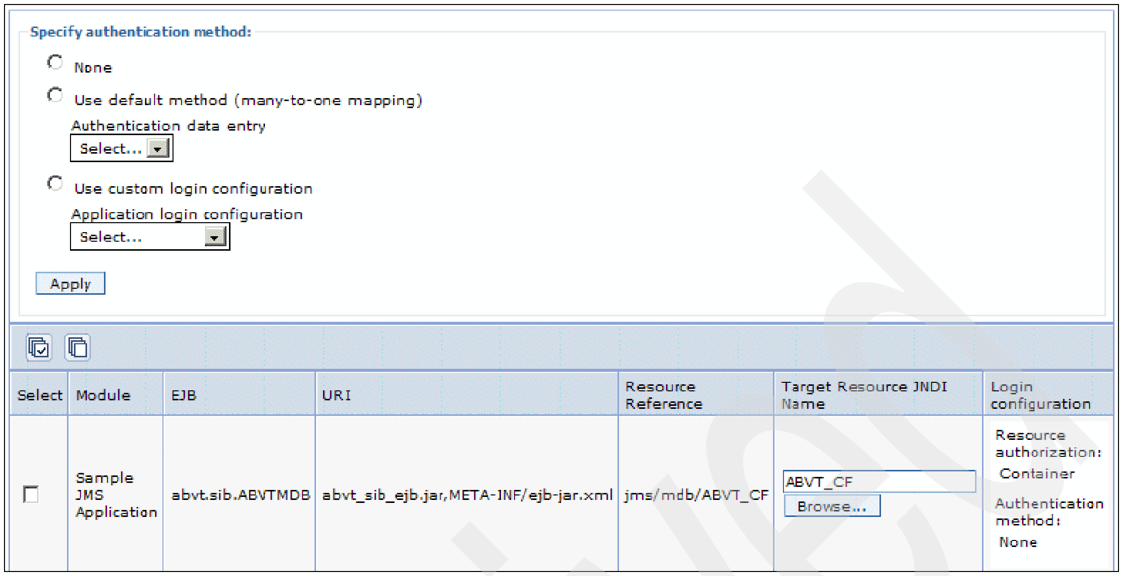
- None
- Use default method (many-to-one mapping)
  - Authentication data entry: Select...
- Use custom login configuration
  - Application login configuration: Select...

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	Sample JMS Application	abvt.sib.ABVTMDB	abvt_sib_ejb.jar,META-INF/ejb-jar.xml	jms/mdb/ABVT_CF	ABVT_CF	Resource authorization: Container Authentication method: None

Figure 25-1 Manage resource references

Resolving the problem when using container authentication

Figure 25-2 on page 556 shows an example where no authentication alias has been configured.



Specify authentication method:

- ○ None
- ○ Use default method (many-to-one mapping)

  Authentication data entry: Select...
- ○ Use custom login configuration

  Application login configuration: Select...

Apply

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	Sample JMS Application	abvt.sib.ABVTMDB	abvt_sib_ejb.jar,META-INF/ejb-jar.xml	jms/mdb/ABVT_CF	ABVT_CF Browse...	Resource authorization: Container Authentication method: None

Figure 25-2   Configuring Container-managed authentication

To configure the authentication alias to be used:

1. Select the check box for the resource reference.
2. Select **Use default method (many-to-one mapping)** and select an authentication alias from the pull-down list.
3. Click **Apply** to make the change to the resource reference.
4. Click **OK**.

Resolving the problem when using application authentication

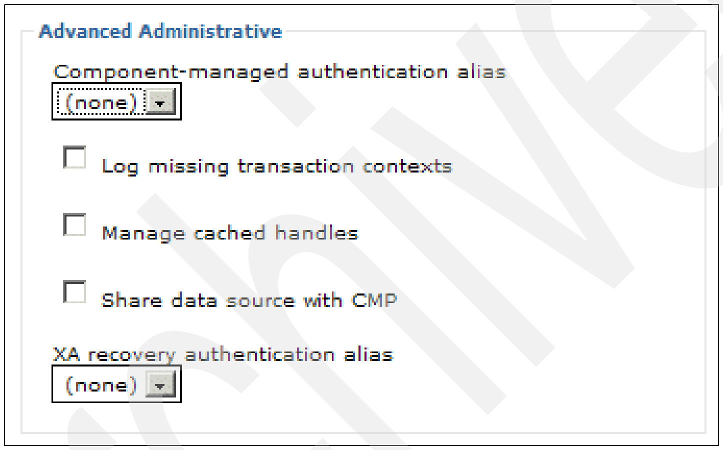
Application authentication is designed for cases in which the application is providing the credential. The application typically provides the credentials in the code. Often, these credentials are thought of as hard coded, although the application might obtain them through some application specific mechanism. If this is the case, consult with the application author to resolve this problem.

Alternatively, WebSphere Application Server provides an additional convenience mechanism for providing credentials called the *component-managed authentication alias*. It is specified on the JMS connection factory and is only used if application authentication is specified and the zero arguments variant of `createConnection()` is used.

It is not good practice to rely on a component-managed authentication alias. The authentication alias is shared by all users of the connection factory who are using application authentication (and direct JNDI lookups).

If you want to supply the credentials on the JMS connection factory, use container-managed authentication instead.

If you are relying on the component-managed authentication alias, you can configure it on the connection factory definitions. You can work with one of these definitions on the menu by expanding **Resources** → **JMS** and selecting one of the connection factories: queue or topic. If the connection factory is for the default messaging provider, look to the bottom of the connection factory definition page at a section titled **Component-managed authentication alias**, and select an authentication alias from the drop-down box; for example, see Figure 25-3.


Advanced Administrative

Component-managed authentication alias: (none)

☐ Log missing transaction contexts

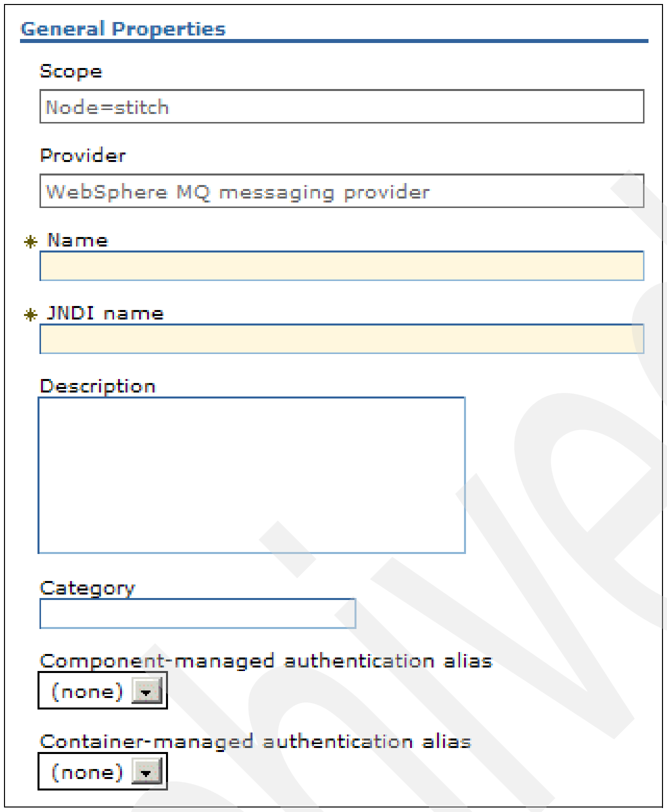
☐ Manage cached handles

☐ Share data source with CMP

XA recovery authentication alias: (none)

Figure 25-3 Default messaging provider component-managed alias

If the connection factory is for the WebSphere MQ messaging provider, use the configuration setting near the top of the properties titled **Component-managed authentication alias**; for example, see Figure 25-4 on page 558.


General Properties

Scope

Node=stitch

Provider

WebSphere MQ messaging provider

* Name

* JNDI name

Description

Category

Component-managed authentication alias

(none)

Container-managed authentication alias

(none)

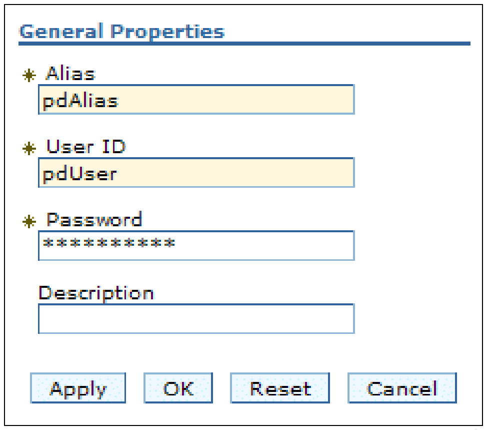
Figure 25-4 WebSphere MQ messaging provider component-managed alias

Note: The use of component-managed authentication aliases is strongly discouraged. Using it is a sign that the application performing the lookup should be using a resource-reference with container authentication and an application update should be performed instead.

Verifying the authentication alias user ID and password

If an authentication alias has been configured (and it is the correct one), you next need to check the configuration of the authentication alias:

1. Select **Security** → **Secure administration, applications, and infrastructure.**
2. Java Authentication and Authorization Service, and select **J2C authentication data**. This page shows a list of authentication aliases.
3. Select the alias that was being used. You see a window similar to Figure 25-5 on page 559.


General Properties

* Alias: pdAlias
* User ID: pdUser
* Password: **********

Description:

Figure 25-5 Configure authentication alias

4. Correct the user name and password, click **OK** and save the configuration.
5. Restart the application server.

   The correct credentials will be used to connect to the messaging provider.

25.4.2 Resolving the problem in a client application

The credentials for a client application are typically provided by calling the `createConnection()` method with a user ID and password. How these are provided is application specific. Contact the application author for further guidance to diagnose this problem.

25.4.3 Validate solution

To validate the solution to your security problem rerun the application. If you made changes to the WebSphere Application Server JMS or JNDI resources, you need to restart the application server processes for changes to take effect.

25.5 Authorization problems connecting to the bus

The primary reason for an authorization failure when connecting to the bus is that the user, or a group that includes the user, does not have the bus connector role for that bus.

You can configure a user to have the bus connector role using the administrative console or the **wsadmin** scripting tool.

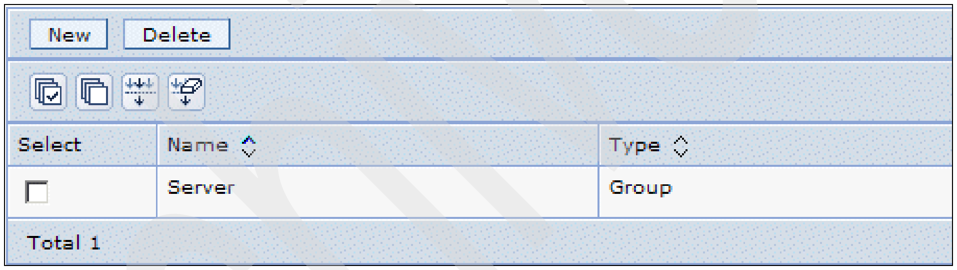
Best practice is to assign roles using groups rather than users directly.

Resolving the problem using the administrative console

From the administrative console:

1. Select **Service Integration** → **Buses**.
2. Click on the bus name to which the client requires access.
3. Select **Security**.
4. Select **Users and groups in the bus connector role**.

   This page lists the users and groups in the bus connector role. By default, this list only contains a single group, the Server group. This convention gives application servers authority to connect to the bus, but not the individual applications . The default is shown in Figure 25-6.



Select	Name	Type
☐	Server	Group

Total 1

Figure 25-6 List of users and groups in bus connector role

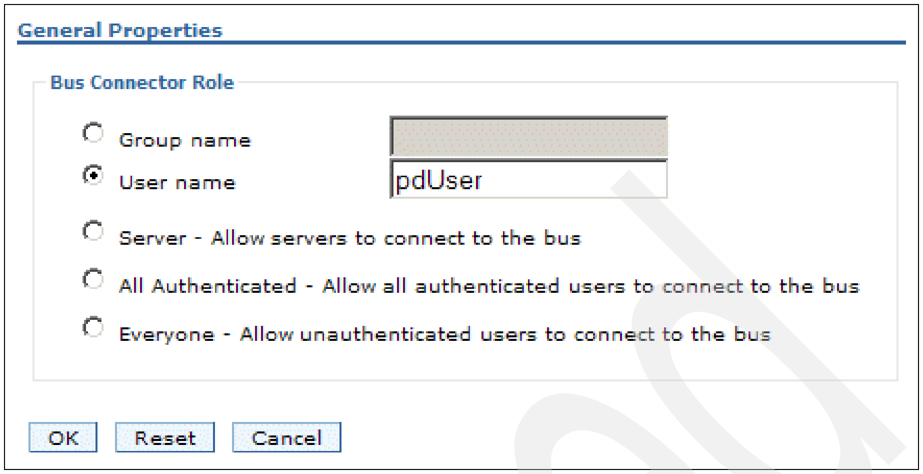
   You can modify this list by either deleting or adding users or groups.

5. To give a user the bus connector role, click **New**.

   A new panel displays that lets you add users and groups (along with the special Everyone, AllAuthenticated, and Server groups) to the bus connector role.

   For illustration purposes in this example, user pdUser is granted direct access to the bus.

   The settings are shown in Figure 25-7 on page 561.

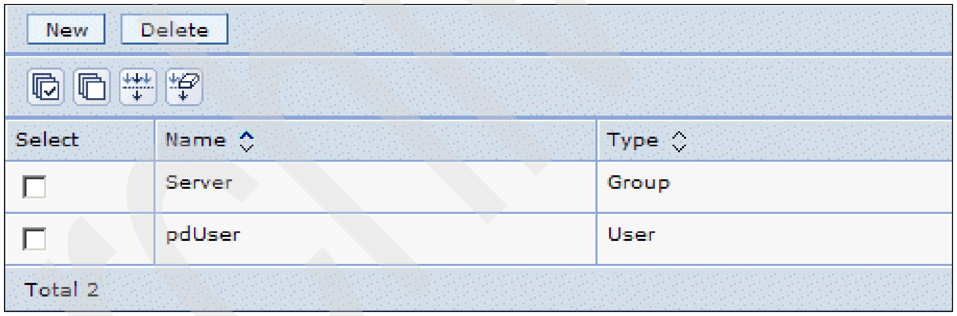

General Properties

Bus Connector Role

( ) Group name
(•) User name: pdUser
( ) Server - Allow servers to connect to the bus
( ) All Authenticated - Allow all authenticated users to connect to the bus
( ) Everyone - Allow unauthenticated users to connect to the bus

Figure 25-7 Adding a user to the bus connector role

6. Click **OK** to add the user to the bus connector role. The result is shown in Figure 25-8.



Select	Name	Type
☐	Server	Group
☐	pdUser	User

Total 2

Figure 25-8 List of users and groups in the bus connector role

7. Save and synchronize the configuration. You do not need to restart the server restart.

Resolving the problem using the wsadmin scripting tool

Alternatively, you can use the **`wsadmin`** scripting tool to configure a user for the bus connector role. There are an extensive set of commands for configuring bus security.

For information about commands for configuring the bus connector role, see the WebSphere Application Server version 6.1 Information Center, *Administering bus connector roles* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/tasks/tjr0100_.html

To modify the bus connector role, load the **wsadmin** tool and enter:

(Using Jacl) `$AdminTask addUserToBusConnectorRole {-bus pdBus -user pdUser}`

(Using Jython) `AdminTask.addUserToBusConnectorRole('[-bus pdBus -user pdUser]')`

25.6 Authorization problems accessing a destination

The primary reason for an authorization failure on a destination is that the user is not in the relevant role for the destination being accessed. A user is in a role if he or she, or a group to which the user belongs, has been configured with that role.

There are several roles that allow types of access. These are shown in Table 25-1.

Table 25-1 Messaging roles and description

Role	Description
Bus connector	Role required to permit the user to connect to the messaging bus.
Creator	Role required to permit the user to create temporary destinations within the name space.
Sender	Role required to permit the user to send a message to a resource (e.g. a Queue, a Foreign bus)
Receiver	Role required to permit the user to receive a message from a resource, for example, a queue or topic.
Browser	Role required to permit the user to browse a message on a resource for example, a queue or topic.

Table 25-2 on page 563 shows which resources the roles are applicable to. The resources shown are not destinations; they are resource types used in the **wsadmin** commands. The Queue resource type is also used for administering roles for Port, Web service, and temporary destination prefixes.

Table 25-2 Which roles are applicable to what resources

	Local Bus	Foreign Bus	Queue	Topic Space	Alias	Foreign Dest	Default	Topic Space Root	Topic
Bus connector	X								
Sender		X	X	X	X	X	X	X	X
Receiver			X	X	X		X	X	X
Browser			X		X		X		
Creator			X				X		

Resolving the problem using the wsadmin scripting tool

The *wsadmin* tool is the only way to administer destination security roles. For detailed instructions on the commands and how they are used, see the Information Center topic *Administering authorization permissions* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/tasks/tjr0380_.html

Best practice is to assign roles using groups rather than users directly. However, for illustration purposes, in the following examples, the user `pdUser` is granted access to the bus.

To resolve a problem *sending* messages, use this command:

```
$AdminTask addUserToDestinationRole {-type Queue -bus pdBus
                      -destination pdQueue -role Sender -user pdUser}
```

To resolve the problem *receiving* messages, use this command:

(Using Jacl) `$AdminTask addUserToDestinationRole {-type Queue -bus pdBus -destination pdQueue -role Receiver -user pdUser}`

(Using Jython) `AdminTask.addUserToDestinationRole('[-type Queue -bus pdBus -destination pdQueue -role Receiver -group pdPUser]')`

If using groups, the commands are similar. For example:

(Using Jacl) `$AdminTask addGroupToDestinationRole {-type Queue -bus pdBus -destination pdQueue -role Receiver -group pdPeople}`

(Using Jython) `AdminTask.addGroupToDestinationRole('[-type Queue -bus pdBus -destination pdQueue -role Receiver -group pdPeople]')`

25.6.1 Validate the solution

To validate the solution to your security problem, rerun the application. If you made changes to the WebSphere Application Server JMS or JNDI resources, you need to restart the application server processes for changes to take effect.

26

Mediation problem determination

Various problems can occur during the mediation of messages within the messaging bus of the WebSphere Application Server. If you experience problems with the mediation of messages, use this chapter to diagnose the problem.

This chapter walks you through the process of debugging problems with the mediation of messages.

26.1 Introduction to mediation

Service integration bus destinations can be configured as mediated destinations. When this is done, a new mediation point is associated with the destination.

The mediation process is:

1. When a message is sent to a mediated destination it is added to the mediation point.
2. The messaging engine takes the message from the mediation point and passes it to a mediation.
3. After the mediation has completed, the message is, as directed by the mediation, processed in one of these ways:
   – Added to the destination
   – Deleted
   – Sent to another destination
   – Sent to the exception destination associated with the destination

This process happens without any information logged to the application server SystemOut log, unless the implementation of the mediation logs it.

For more information about message mediation see the WebSphere Application Server V6.1 Information Center topic *Mediations* at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.pmc.nd.doc/tasks/tjp9999_.html`

26.2 Identify symptoms

Mediation problems discussed are categorized by their major symptom. The first step is to determine which of the following symptoms fit your problem:

- Messages are not being consumed by the application.
- Messages are being consumed, but are unmediated.
- Messages are mediated incorrectly.
- Messages are mediated, but slowly.

26.3 Messages are not being consumed by the application

There are two primary reasons that an application does not consume mediated messages:

- Messages are queued on the correct destination but are waiting to be mediated.
- Messages are being sent to the wrong destination.

26.3.1 Determine if messages are queued on the mediation point

First, determine if messages are queued on the mediation point waiting to be mediated. To examine the current message depth at the mediation point, using the administrative console:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Destinations**.
4. Click on the destination for the messages.
5. Click **Mediation Points**.
6. Click on the relevant mediation point (you might need to examine each of the mediation points in turn).
7. Click the **Runtime** tab.
8. Examine the message depth to see if it looks unusually large or is increasing faster than the messages are being handled.

If the message depth continually increases with no indication any messages are being processed, see 26.6, “Messages are waiting to be mediated” on page 569.

If the message depth both increases and decreases, but messages are arriving faster than they are being handled, see 26.8, “Messages are mediated, but slowly” on page 577.

26.3.2 Determine if messages are queued at the wrong destination

If no messages appear queued to the mediation point and they are not being consumed by the application, the messages might be at the wrong destination, because either the mediation put them on the wrong destination or the messages are at an exception destination.

If you have this situation, see 26.9, “Messages are arriving at the wrong destination” on page 579.

26.4 Messages are being consumed, but are unmediated

If messages are being consumed by an application, but the results of that message consumption indicate that:

- The required message *mediation is not being performed*, see 26.5, “Mediation is not performed” on page 568.
- The required message *mediation is being performed incorrectly*, see 26.7, “Messages are mediated incorrectly” on page 574.

If messages are being mediated correctly, but the *arrival rate of messages at that destination is lower than expected*, see 26.8, “Messages are mediated, but slowly” on page 577.

26.5 Mediation is not performed

If messages are arriving at the application unmediated:

- Verify the destination is mediated.
- Verify the message is being routed to the mediated destination.

26.5.1 Verify the destination is mediated

Use the administrative console to verify the configuration of the relevant destinations:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Destinations**.
   This page lists the destinations defined on the bus and shows any mediations associated with the destination.

Solution

If no mediation is associated with the destination, you need to configure it. Select the destination and click **Mediate**.

26.5.2 Verify the message is being routed to the mediated destination

The use of *forward routing paths* is a common reason that a mediation is not performed. The forward routing path is a property of the message that can be used by the producing application to determine a route for the message. As a result, a message, once in the system, might be routed differently than expected. A mediation can be used to manipulate the forward and reverse routing paths of a message.

For more information about forward routing path capabilities see *Destination routing paths* at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.pmc.nd.doc/concepts/cjo0005_.html`

Solution

Ensure the correct destination is being mediated and that the routing path is correct for your configuration.

The mediated destination is usually the destination from which the application is expecting to consume messages. For a more complex application design that uses the forward routing path capabilities, you need to examine the application design to determine which destinations are mediated.

One option is to inject a message and trace it through the system. An alternative is to use the administrative console to stop the message points, starting them one at a time to watch the message manually traverse the path.

If you make changes to application server resources, you need to restart the application server for those changes to take effect.

26.6 Messages are waiting to be mediated

If messages are not being consumed by any application but appear to be waiting to be mediated, collect and analyze the diagnostic data to determine what is causing the messages to wait.

26.6.1 Verify the mediation point is operational

You can use the administrative console to verify that the mediation point is operational (that is, that messages at the mediation point will be passed to the mediation code to be mediated):

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Destinations**.
4. Click on the destination name.
5. Click **Mediation Points** in the Message points.

   This page lists the mediation points of the destination and shows their current status, which should `Started`.

   If it is any other state, attempt to start it by selecting the mediation point and clicking **Start**.

There are several reasons the mediation point might be stopped when there are no problems:

- The mediation point was stopped by an operator and not restarted.
- The initial state has been configured as `Stopped`.
- The mediation point has been deleted and then redefined. The previous deletion has not yet completed. After the previous deletion has completed, the mediation point will automatically start when the server has started, assuming that the configured initial state for the new mediation point is `Started`.

If the mediation point was stopped and you were able to start it, verify the messages are flowing to the application.

If the mediation point was stopped and you are not sure why it was stopped, or if it fails to start, continue to 26.6.2, "Verify the mediation application is started" on page 570 to collect diagnostics.

If the mediation point was started and this does not appear to be the reason messages are waiting, continue to 26.6.2, "Verify the mediation application is started" on page 570 to collect diagnostics.

26.6.2 Verify the mediation application is started

To view the status of the application select **Applications** → **Enterprise applications** in the administrative console. If the application is not started, start it by selecting the box to the left of the application and clicking **Start**.

26.6.3 Analyze SystemOut

If the previous system checks have not resolved the problem, you need to analyze the SystemOut log for the bus member to which the queue and mediation points have been assigned. If the bus member is a cluster, you might need to repeat these steps for each server in the cluster.

If the mediation point is not in the `Started` state, messages are not being processed. You should examine the SystemOut log file for the following messages:

- **`CWSIP0771I:`** `Administrator action`

  An administrator has manually stopped the mediation point. Start the mediation point manually.

- **`CWSIP0773I:`** `Using configured initial state`

  The mediation point has been configured with an initial state of `Stopped`. Either start the mediation point manually, or change the initial state of the mediation point to be `Started` and restart the messaging engine or application server.

- **`CWSIP0775I:`** `Waiting for the deletion of another mediation point on this destination.`

  The mediation point has been deleted and then redefined. The previous deletion has not yet completed. After the previous deletion has completed the mediation point will automatically start when the server has started, assuming that the configured initial state for the new mediation point is `Started`.

- **`CWSIZ0002E:`** `The mediation named` *`Mediation_Name`* `that is attached to destination` *`Destination_Name`* `is defined to use mediation handler list` *`Mediation_Handler_List_Name`*`. However this handler list does not exist.`

  See 26.6.4, "Mediation is not configured correctly" on page 573.

- **`CWSIZ0011E:`** `The mediation named` *`Mediation_Name`* `attached to destination` *`Destination_Name`* `could not process the message` *`Message_ID`* `because the application` *`Mediation_Application_Name`* `has not started.`

  Start the application and the mediation point will automatically start to process messages.

- **`CWSIZ0020E:`** `The mediation authentication alias for the bus called` *`Bus_Name`* `could not be located.`

  or

`**CWSIZ0021E:**` The mediation authentication alias mediator for the bus called *Bus_Name* could not be resolved.

or

CWSIZ0045E: `The mediation authentication alias mediator for the bus called` *Bus_Name* `could not be authenticated.`

See 26.6.5, “Invalid mediation authentication alias” on page 573.

- **CWSIZ0056E:** `The mediation named` *Mediation_Name* `attached to destination` *Destination_Name* `could not process the message` *Message_ID* `because the next destination` *Next_Destination* `cannot accept the message due to exception com.ibm.ws.sib.processor.exceptions.SIMPLimitExceededException.`

  The message, after the mediation code completed, has a forward routing path. However the message could not be sent along that forward routing path. The mediated message was not delivered to its next destination due to problems with the next destination.

  See Chapter 16, “Messaging in a multiple messaging engine environment” on page 429 for more problem determination guidance.

- **CWSIZ0057E:** `The mediation named` *Mediation_Name* `attached to destination` *Destination_Name* `could not process the message` *Message_ID* `because the message cannot be made available to consumers due to exception ...`

  The mediation has completed successfully, but when the system attempted to make the message available to the consumers it was unable to do so. The exception information should provide more information about the problem. Resolve any problems that might exist with the next destination and restart the mediation.

- **CWSIZ0058E:** `The mediation named` *Mediation_Name* `attached to destination` *Destination_Name* `could not process the message` *Message_ID* `because the connection to the messaging engine has been lost as reported by exception com.ibm.ws.sib.jfapchannel.JfapConnectionBrokenException.`

  The mediation has been configured to use a mediation execution point that is remote from the queue point of the destination. The mediation has completed successfully; however, when the system attempted to return the message to the messaging engine, the system reported that the connection was no longer available.

  The most common problem is that the messaging engine is unavailable or is stopping. Validate that all messaging engines are in the correct state. Resolve the problem and restart the mediation.

- **CWSIZ0059E:** `The mediation named` *`Mediation_Name`* `that is attached to destination` *`Destination_Name`* `is defined to use mediation handler list` *`Mediation_List_Name`*`. However no handler lists exist.`

  See 26.6.4, “Mediation is not configured correctly” on page 573.

If none of these symptoms apply, but it appears that the mediation is not being performed, the mediation might be running slowly. To investigate this, see 26.8, “Messages are mediated, but slowly” on page 577.

26.6.4 Mediation is not configured correctly

Use the administrative console to verify that the mediation has been configured correctly:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Mediations** in the Destination resources.
4. Click on the mediation name.

Pay specific attention to the mediation handler list name. This case sensitive name must match exactly the handler list name that was defined when the mediation was developed. When the handler list cannot be found, but other handler lists can be located, the system generates a **`CWSIZ0052I`** message which is placed in the logs for the relevant server. You can use this message to verify the spelling of the handler list name.

An example of this message is:

```
CWSIZ0052I: Mediation handler lists defined in the server are
[SimpleMediationHandler].
```

If the mediation appears to be correctly configured but the mediation is not being performed, see 26.5, “Mediation is not performed” on page 568.

26.6.5 Invalid mediation authentication alias

Use the administrative console to verify that the mediation authentication alias has been configured correctly:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Security** in the Additional Properties.

Check that the correct authentication alias has been selected and that the user ID and password for that authentication alias are correct.

26.6.6 Validate the solution

If you made changes to the application server resources, restart the application server so that the changes take effect. To validate the solution to your mediation problem, send a test message to the system and verifying that it was correctly mediated.

26.7 Messages are mediated incorrectly

If messages are being consumed by an application but the results of that message consumption indicate that the required message mediation is being incorrectly performed, you need to trace the message mediation to determine if the mediation code is operating as expected.

26.7.1 Collect diagnostics

To verify that the mediation code is operating as expected, take a trace to determine how the mediation modifies the message and what action is taken with the message after the mediation has completed.

The trace must be for the server of the bus member to which the queue and mediation points have been assigned. If the bus member is a cluster, you might need to repeat these steps for each server in the cluster.

Using the administrative console:

1. Select **Troubleshooting** → **Logs and Traces** in the navigation bar.
2. Click on the name of the server where the mediation is running.
3. Click **Change Log Details Levels**.
4. Select the **Runtime** tab.
5. Add `SIBMessageTraceContentsMediations=all` to the Groups in the large text box. (Use a colon (`:`) to separate clauses in this box.)
6. Click **Apply**.

Recreate the problem; then, turn off the trace by resetting the trace string.

26.7.2 Analyze the trace

By default, the trace is stored in ${SERVER_LOG_ROOT}/trace.log.

The trace log contains records similar to Example 26-1.
Note: The timestamps in the example have been replaced with ellipsis (…).

Example 26-1 Mediation trace

```
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus] CWSIZ1000I: A message
with ID ID:c0aa5975e6450404189fb1e1110a134f0000000000000001 and System
Message ID 56F8B38EDBCBB8E9_500003 has been delivered to mediation
Modifying attached to destination Modify.
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] Discriminator
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] ForwardRoutingPath
= []
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] ReverseRoutingPath
= []
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] Priority
= 4
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] Reliability
= ReliablePersistent
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] TimeToLive
= 0
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] RemainingTimeToLive
= -1
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] ReplyDiscriminator
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] ReplyPriority
= 4
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] ReplyReliability
= None
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] ReplyTimeToLive
= 0
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] RedeliveredCount
= 0
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] Userid
=
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMSType
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMSXAppID
= Service Integration Bus
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMSXDeliveryCount
= 1
```

```
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMSXGroupID
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMSXGroupSeq
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Format
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Feedback
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_PutApplType
= null
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_Report_Exception            = null
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_Report_Expiration           = null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Report_COA
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Report_COD
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Report_PAN
= false
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Report_NAN
= false
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_Report_Pass_Msg_ID          = false
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_Report_Pass_Correl_ID       = false
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_Report_Discard_Msg          = false
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_Last_Msg_In_Group           = null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_PutDate
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_PutTime
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Encoding
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_Character_Set
= null
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] JMS_IBM_ExceptionReason
= null
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_ExceptionMessage            = null
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_ExceptionTimestamp          = null
```

```
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus]
JMS_IBM_ExceptionProblemDestination = null
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus] Format = JMS:text
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus] (DataGraph) {
(DataObject: JmsTextBody) {
    data=(DataObject: DataType1) {
      value=(Data)'test'
    }
  }}
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus] CWSIZ1001I: A message
with ID ID:c0aa5975e6450404189fb1e1110a134f0000000000000001 and System
Message ID 56F8B38EDBCBB8E9_500003 has been sent on the forward routing
path [] by mediation Modifying attached to destination Modify.
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] TimeToLive
= 10000
[…] 00000034 MediationMess 3  [MyBus:s1-MyBus] RemainingTimeToLive
= 10000
[…] 00000034 MediationMess 3   [MyBus:s1-MyBus] level = gold
```

The trace log shows the contents of the message before the mediation executes, the action that was taken after the mediation completed, and, if the message was not deleted, what the mediation changed in the message.

Solution

Review the entries to ensure that the actions taken are what you expect. If not, correct and redeploy the mediation application.

26.7.3 Validate the solution

To validate the solution to your mediation problem, send a test message to the system and verify that it was correctly mediated.

26.8 Messages are mediated, but slowly

Messages are being consumed by an application, but the arrival rate of messages at that destination is lower than expected.

26.8.1 Configure for performance

A mediation will be a performance bottleneck if the mediation is unable to process messages as fast as the messages are delivered to the mediation point.

When this bottleneck occurs, the message depth at the mediation point will steadily increase as messages are queued waiting for the mediation to run.

If the mediation point seems to be a bottleneck for traffic, check the configuration to ensure you have taken advantage of performance features.

Allow concurrent message mediation

If the mediation was designed so that different instances of the mediation can be working on different messages simultaneously (and message ordering is not a requirement), the mediation can be configured to permit this behavior. To verify that the mediation has been configured to permit concurrent message mediation, in administrative console:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Mediations** in the Destination Resources.
4. Click on the relevant mediation.
5. Ensure that the **Allow concurrent mediation** box is selected to permit concurrent mediation.

Increase thread pool size

If concurrent mediation is permitted and the mediation is still a bottleneck but there is spare capacity on the server where the mediation is running, you can increase the size of the mediation thread pool to permit more concurrent mediations to run. To examine and modify the mediation thread pool in the administrative console:

1. Select **Service Integration** → **Buses** in the navigation bar.
2. Click on the bus name.
3. Click **Messaging Engines** in Topology.
4. Click on the relevant messaging engine. You might need to examine each messaging engine in turn.
5. Click **Mediation thread pool** in the Additional properties.

26.8.2 Validate the solution

Restart the server and send test messages to the system. Verify that these are correctly mediated with improved throughput.

26.9 Messages are arriving at the wrong destination

If messages arrive at the wrong destination, you need to investigate why the messages arrived at that destination.

26.9.1 Examine the messages

To find messages that might be arriving at the wrong destination:

1. Stop any applications that might be consuming the messages.
2. Send a test message.
3. Use the administrative console to find and examine the message.
4. Select **Service Integration** → **Buses** in the navigation bar and select the bus.
5. Click **Destinations**. For each destination that could have the message:
   a. Click on the destination name.
   b. Click **Queue Points** and then click the queue point name.
   c. Click the Runtime tab.
   d. Click **Messages.** If the message is there, click on the message.

      The **exception destination reason** field of the message is either blank or it contains a message:

      - If the exception destination reason field is *blank*, the mediation (or sending application) has explicitly sent the message to the wrong destination.
      - If the **exception destination reason** field is *not empty*, the message has been sent to the destination because a problem was encountered, and this destination is defined as the exception destination.

        Examine the reason and see Exception destination reasons to determine why the message was rerouted.

Exception destination reasons

The reason codes provide information about the problem; either something is wrong with the mediation code or something is wrong with the message itself. Use this information to decide whether you need to examine the mediation logic or message, and what to look for. If the problem is with the message, trace the mediation to see what is happening. For information about tracing the mediation, see 26.7, “Messages are mediated incorrectly” on page 574:

- **`CWSIK0101W:`** `The message` *`Message_ID`* `has been re-routed to the exception destination` *`Exception_Destination_Name`* `because the mediation` *`Mediation_Name`* `at destination` *`Destination_Name`* `caused the exception java.lang.NullPointerException.`

  The mediation sent the message to the exception destination.
- **`CWSIK0102E:`** `The message` *`Message_ID`* `cannot progress further along its forward routing path after being mediated by mediation` *`Mediation_Name`* `at destination` *`Destination_Name`* `because the message does not conform to the message format JMS:text.`

  The mediation modified the message in such a way that the contents of the message no longer matches the schema of the message.
- **`CWSIK0103E:`** `The message` *`Message_ID`* `cannot progress further along its forward routing path after being mediated by mediation` *`Mediation_Name`* `at destination` *`Destination_Name`* `because the message is not well formed.`

  The mediation modified the message in such a way that that the message can no longer be serialized.

26.10 Search online support

If you are sure the problem is in the mediation process, there are tasks that you can do before contacting IBM support. First, review the documentation that you have gathered for errors that were not addressed in this paper and search support sites for information or fixes. Look for current information available from IBM support on known issues and resolutions on this IBM support page:

`http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&q=mediation&Go.x=0&Go.y=0`

27

The next step

This chapter contains information about online resources that you will find useful in diagnosing problems and includes links to the MustGather information that you will need before contacting IBM technical support.

27.1 Online resources

The symptoms and problem areas included in this activity are some that you are more likely to experience. However, there are other things that can go wrong. If you are still experiencing a problem, you can use the following resources to help debug the problem further:

- Further explanation and description of appropriate user actions can be found by searching for each error message code in the WebSphere Application Server, version 6.1 Information Center.

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

27.1.1 WebSphere MQ resources

If you were unable to resolve the problem using the instructions above then you can use the following resources to help debug the problem further.

- The WebSphere MQ library can be found at:

  http://www-306.ibm.com/software/integration/wmq/library/

  – The WebSphere MQ System Administration Guide contains information about examining the WebSphere MQ error logs.

  – the WebSphere MQ Messages Reference contains a detailed description of WebSphere MQ reason codes and suggested actions

- WebSphere MQ reason codes can be found at:

  http://www-306.ibm.com/software/integration/mqfamily/library/manualsa/amqzao/amqzao0m.htm

27.2 Contact IBM

Before contacting IBM technical support you should examine the relevant MustGather documents and collect the appropriate documentation for your problem.

For problems using the *default messaging provider* see:

- Mustgather: Service Integration Technology

  http://www-1.ibm.com/support/docview.wss?uid=swg21266769

For problems using the WebSphere MQ JMS provider see the following:

- MustGather: Documentation required by WebSphere MQ - Java and JMS problems

  `http://www-1.ibm.com/support/docview.wss?uid=swg21176943`

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks Publications

For information about ordering these publications, see "How to get IBM Redbooks publications" on page 590. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- *Architecting High Availability Using WebSphere V6 on z/OS*, SG24-6850
- *WebSphere for z/OS Problem Determination Means and Tools*, REDP-6880
- *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257
- *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304
- *WebSphere Application Server V6.1: Planning and Design*, SG24-7305
- *Approach to Problem Determination in WebSphere Application Server V6*, REDP-4073
- *WebSphere Application Server V6: Web Server Plug-in Problem Determination*, REDP-4045
- *WebSphere Application Server V6.1: Workload Management Problem Determination*, REDP-4308

Online resources

These Web sites are also relevant as further information sources:

- The IBM Guided Activity Assistant

  `http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24011818`

DevelopersWorks

- *Configure a Service Integration Bus in a network deployment environment*

  http://www-128.ibm.com/developerworks/webservices/library/ws-sibus/

- *Configure platform messaging and Web Services Gateway for a clustered environment, Part 1: Cluster enhancements for WebSphere Application Server Version 6.1 on z/OS*

  http://www-128.ibm.com/developerworks/webservices/library/ws-cluster env1/index.html

Specifications

- JSR-000101 JavaTM APIs for XML based RPC:

  http://jcp.org/aboutJava/communityprocess/final/jsr101/index2.html

- *JSR-000152 JavaServer Pages 2.0 Specification*, for issues related to internationalization:

  http://jcp.org/aboutJava/communityprocess/final/jsr152/index.html

- JSR-000154 JavaTM Servlet 2.4 Specification

  http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html

- Sun Developers Network Web Services Reference:

  http://java.sun.com/webservices/reference/apis-docs/index.jsp

- Java API for XML-Based RPC (JAX-RPC) Downloads & Specifications

  http://java.sun.com/xml/downloads/jaxrpc.html

- The JAX-RPC 1.1 specification

  http://developers.sun.com/techtopics/webservices/reference/api/index .html

- JSR 109: Implementing Enterprise Web Services

  http://jcp.org/en/jsr/detail?id=109

- Basic Profile Version 1.1

  http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

- SOAP 1.1

  http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

- SOAP with Attachments API for Java (SAAJ) Specification 1.1

  https://saaj.dev.java.net/

- Web Services Description Language (WSDL) 1.1

  http://www.w3.org/TR/wsdl

- XML Schema

  http://www.w3.org/2001/XMLSchema
- *RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax*

  http://www.faqs.org/rfcs/rfc2396.html

Support articles

- *Customizing SimpleFileServlet*:

  http://www.ibm.com/support/docview.wss?uid=swg21116838
- *Web server plug-in policy for WebSphere Application Server*

  http://www-1.ibm.com/support/docview.wss?uid=swg21160581
- *Understanding IBM HTTP Server plug-in Load Balancing in a clustered environment*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219567
- *Understanding HTTP plug-in failover in a clustered environment*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21219808
- *WebSphere plugin failover delayed when system is physically unavailable*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21052862
- *How to use packet trace tools iptrace, snoop, tcpdump, ethereal, and nettl*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21175744
- *PK20304: NO_IMPLEMENT ON FIRST REQUEST TO A CLUSTER*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg1PK20304
- *Tune High Availability (HA) Manager configuration for large cell environments*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21251873
- *CWRLS0030W message continuously logged and WebSphere Application Server fails to open for e-business*

  http://www.ibm.com/support/docview.wss?uid=swg21245012
- *Tune High Availability (HA) Manager configuration for large cell environments*

  http://www.ibm.com/support/docview.wss?rs=180&uid=swg21251873
- *Memory leak detection and analysis in WebSphere Application Server: Part 2: Tools and features for leak detection and analysis*

  http://www-128.ibm.com/developerworks/websphere/library/techarticles/0608_poddar/0608_poddar.html
- Learning more about the Classloader

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21242692

Product pages (Home, support, infocenter)

- IBM Information Management Software for z/OS Solutions Information Center

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp

- IBM HTTP Server Support

  http://www-306.ibm.com/software/webservers/httpservers/support

- IBM HTTP Server, Version 6.1 Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.ihs.doc/info/welcome_ihs.html

- IBM HTTP Server, Version 6.1 Information Center, *Troubleshooting IBM HTTP Server*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs

- IBM HTTP Server for iSeries Support

  http://www-03.ibm.com/servers/eserver/iseries/software/http/services/service.html

- IBM HTTP Server for iSeries documentation

  http://www-03.ibm.com/servers/eserver/iseries/software/http/docs/doc.html

- WebSphere Application Server V6.1 Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

- WebSphere Application Server Product Support page

  http://www.ibm.com/software/webservers/appserv/was/support/

- WebSphere Application Server V6.1 System Requirements

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007651

- WebSphere Application Server detailed system requirements

  http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

- WebSphere MQ Information Center

  http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp

- WebSphere MQ library

  http://www-306.ibm.com/software/integration/wmq/library/

- WebSphere MQ reason codes

  http://www-306.ibm.com/software/integration/mqfamily/library/manualsa/amqzao/amqzao0m.htm

MustGather

- *MustGather: Read first for IBM HTTP Server*

  http://www-1.ibm.com/support/docview.wss?rs=177&uid=swg21192683

- Index of MustGather documentation

  http://www-1.ibm.com/support/docview.wss?uid=swg21145599

- MustGather: Classloader problems for WebSphere Application Server

  http://www-1.ibm.com/support/docview.wss?uid=swg21196187

- MustGather: Documentation required by WebSphere MQ - Java and JMS problems

  http://www-1.ibm.com/support/docview.wss?uid=swg21176943

- *MustGather: i18n (Internationalization) / Double Byte Character Set (DBCS)*

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21141732

- *MustGather: Java Server Faces (JSF) problems in V6.x*

  http://www.ibm.com/support/docview.wss?uid=swg21198110

- *MustGather: JavaServer Pages (JSP) problems*

  http://www-1.ibm.com/support/docview.wss?uid=swg21255205

- MustGather: Service Integration Technology

  http://www-1.ibm.com/support/docview.wss?uid=swg21199330

- *MustGather: Web container errors on WebSphere Application Server V6.1*

  http://www-1.ibm.com/support/docview.wss?uid=swg21252138

- MustGather: Web services engine and tooling problems for WebSphere Application Server V6.1, V6, V5.1 and V5

  http://www-1.ibm.com/support/docview.wss?rs=180&context=SSCR4XC&q1=MustGatherDocument&uid=swg21198363&loc=en_US&cs=utf-8&lang=en

- MustGather: Web services security (WS-Security) problems with WebSphere Application Server

  http://www-1.ibm.com/support/docview.wss?rs=180&context=SSCNPY4&q1=MustGatherDocument&uid=swg21199335&loc=en_US&cs=utf-8&lang=en

- Fix Central

  http://www-912.ibm.com/eserver/support/fixes/fixcentral/main/iseries/

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks, IBM Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy IBM Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

WebSphere Application Server V6.1 Problem Determination

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection

Identify common symptoms

Collect and analyze the data

Find a resolution

This IBM® Redbooks® publication is a collection of previously published Redpapers. This publication allows for easy download of all papers.

Each paper addresses the problem determination process to take for specific components on specific platforms. The intent of the papers is to help customers through the process of identifying and resolving problems in WebSphere Application Server V6.1. The reader will be taken through the process of identifying symptoms of the problem, collecting and analyzing data for diagnosing the problem, examining common root causes and solutions for a problem, and finally how to gather documentation before contacting IBM technical support.

As a prerequisite to problem determination efforts on WebSphere® Application Server, you should review the following paper:

Approach to Problem Determinations

http://www.redbooks.ibm.com/abstracts/redp4073.html

SG24-7461-00 ISBN 0738485616