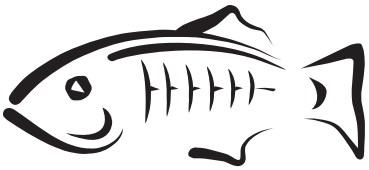


# Open Message Queue

Release Notes

Release 4.5.2

February 2012

This book describes new features, compatibility issues, and existing bugs for the Message Queue 4.5.2 release.

Open Message Queue Release Notes, Release 4.5.2

# Preface

This book provides information about concepts and procedures for developing Java messaging applications (Java clients) that work with Message Queue.

This preface consists of the following sections:



## Documentation Conventions

This section describes the following conventions used in Message Queue documentation:



### Typographic Conventions

The following table describes the typographic conventions that are used in this book.

| Typeface | Meaning | Example |
|---|---|---|
| `AaBbCc123` | The names of commands, files, and directories, and onscreen computer output | Edit your `.login` file.<br>Use `ls a` to list all files.<br>`machine_name% you have mail.` |
| **`AaBbCc123`** | What you type, contrasted with onscreen computer output | `machine_name%` **`su`**<br>`Password:` |
| *aabbcc123* | Placeholder: replace with a real name or value | The command to remove a file is `rm` *filename*. |
| *AaBbCc123* | Book titles, new terms, and terms to be emphasized | Read Chapter 6 in the *User's Guide*.<br>A *cache* is a copy that is stored locally.<br>Do *not* save the file.<br>**Note:** Some emphasized items appear bold online. |

## Symbol Conventions

The following table explains symbols that might be used in this book.

| Symbol | Description | Example | Meaning |
|---|---|---|---|
| `[ ]` | Contains optional arguments and command options. | `ls [-l]` | The `-l` option is not required. |
| `{ \| }` | Contains a set of choices for a required command option. | `-d {y\|n}` | The `-d` option requires that you use either the `y` argument or the `n` argument. |
| `${ }` | Indicates a variable reference. | `${com.sun.javaRoot}` | References the value of the `com.sun.javaRoot` variable. |
| - | Joins simultaneous multiple keystrokes. | Control-A | Press the Control key while you press the A key. |
| + | Joins consecutive multiple keystrokes. | Ctrl+A+N | Press the Control key, release it, and then press the subsequent keys. |
| > | Indicates menu item selection in a graphical user interface. | File > New > Templates | From the File menu, choose New. From the New submenu, choose Templates. |

## Shell Prompt Conventions

The following table shows the conventions used in Message Queue documentation for the default UNIX system prompt and superuser prompt for the C shell, Bourne shell, Korn shell, and for the Windows operating system.

| Shell | Prompt |
|---|---|
| C shell on UNIX, Linux, or AIX | *machine-name*`%` |
| C shell superuser on UNIX, Linux, or AIX | *machine-name*`#` |
| Bourne shell and Korn shell on UNIX, Linux, or AIX | `$` |
| Bourne shell and Korn shell superuser on UNIX, Linux, or AIX | `#` |
| Windows command line | `C:\>` |

## Directory Variable Conventions

Message Queue documentation makes use of three directory variables; two of which represent environment variables needed by Message Queue. (How you set the environment variables varies from platform to platform.)

The following table describes the directory variables that might be found in this book and how they are used. Some of these variables refer to the directory *mqInstallHome*, which is the directory where Message Queue is installed to when using the installer or unzipped to when using a zip-based distribution.

---

**Note:** In this book, directory variables are shown without platform-specific environment variable notation or syntax (such as `$IMQ_HOME` on UNIX). Non-platform-specific path names use UNIX directory separator (`/`) notation.

---

| Variable | Description |
|---|---|
| IMQ_HOME | The Message Queue home directory:<br>■ For installations of Message Queue bundled with GlassFish Server, IMQ_HOME is *as-install-parent*/mq, where *as-install-parent* is the parent directory of the GlassFish Server base installation directory, glassfish3 by default.<br>■ For installations of Open Message Queue, IMQ_HOME is *mqInstallHome*/mq. |
| IMQ_VARHOME | The directory in which Message Queue temporary or dynamically created configuration and data files are stored; IMQ_VARHOME can be explicitly set as an environment variable to point to any directory or will default as described below:<br>■ For installations of Message Queue bundled with GlassFish Server, IMQ_VARHOME defaults to *as-install-parent*/glassfish/domains/domain1/imq.<br>■ For installations of Open Message Queue, IMQ_HOME defaults to *mqInstallHome*/var/mq. |
| IMQ_JAVAHOME | An environment variable that points to the location of the Java runtime environment (JRE) required by Message Queue executable files. By default, Message Queue looks for and uses the latest JDK, but you can optionally set the value of IMQ_JAVAHOME to wherever the preferred JRE resides. |

# Related Documentation

The information resources listed in this section provide further information about Message Queue in addition to that contained in this manual. The section covers the following resources:

- Message Queue Documentation Set
- Java Message Service (JMS) Specification
- JavaDoc
- Example Client Applications
- Online Help

## Message Queue Documentation Set

The documents that constitute the Message Queue documentation set are listed in the following table in the order in which you might normally use them. These documents are available through the Oracle GlassFish Server documentation web site at http://www.oracle.com/technetwork/indexes/documentation/index.html.

| Document | Audience | Description |
|---|---|---|
| *Technical Overview* | Developers and administrators | Describes Message Queue concepts, features, and components. |
| *Release Notes* | Developers and administrators | Includes descriptions of new features, limitations, and known bugs, as well as technical notes. |
| *Administration Guide* | Administrators, also recommended for developers | Provides background and information needed to perform administration tasks using Message Queue administration tools. |

| Document | Audience | Description |
|---|---|---|
| *Developer's Guide for Java Clients* | Developers | Provides a quick-start tutorial and programming information for developers of Java client programs using the Message Queue implementation of the JMS or SOAP/JAXM APIs. |
| *Developer's Guide for C Clients* | Developers | Provides programming and reference documentation for developers of C client programs using the Message Queue C implementation of the JMS API (C-API). |
| *Developer's Guide for JMX Clients* | Administrators | Provides programming and reference documentation for developers of JMX client programs using the Message Queue JMX API. |

## Java Message Service (JMS) Specification

The Message Queue message service conforms to the Java Message Service (JMS) application programming interface, described in the *Java Message Service Specification.* This document can be found at the URL `http://www.oracle.com/technetwork/java/jms/index.html`.

## JavaDoc

JMS and Message Queue API documentation in JavaDoc format is included in Message Queue installations at `IMQ_HOME/javadoc/index.html`. This documentation can be viewed in any HTML browser. It includes standard JMS API documentation as well as Message Queue-specific APIs.

## Example Client Applications

Message Queue provides a number of example client applications to assist developers.

### Example Java Client Applications

Example Java client applications are included in Message Queue installations at `IMQ_HOME/examples`. See the `README` files located in this directory and its subdirectories for descriptive information about the example applications.

### Example C Client Programs

Example C client applications are included in Message Queue installations at `IMQ_HOME/examples/C`. See the `README` files located in this directory and its subdirectories for descriptive information about the example applications.

### Example JMX Client Programs

Example Java Management Extensions (JMX) client applications are included in Message Queue installations at `IMQ_HOME/examples/jmx`. See the `README` files located in this directory and its subdirectories for descriptive information about the example applications.

## Online Help

Online help is available for the Message Queue command line utilities; for details, see "Command Line Reference" in *Open Message Queue Administration Guide*. The Message Queue graphical user interface (GUI) administration tool, the Administration Console, also includes a context-sensitive help facility; for details, see "Administration Console Online Help" in *Open Message Queue Administration Guide*.

## Documentation, Support, and Training

The Oracle web site provides information about the following additional resources:

- Documentation (http://www.oracle.com/technetwork/indexes/documentation/index.html)
- Support (http://www.oracle.com/us/support/044752.html)
- Training (http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=315)

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

### Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

# 1

# Release Notes

These release notes contain important information available at the time of release of Message Queue 4.5.2. New features and enhancements, known issues and limitations, and other information are addressed here. Read this document before you begin using Message Queue 4.5.2.

These release notes also contain information about the 4.5.1, 4.5, 4.4.2, 4.4 Update 1, 4.4, 4.3, 4.2, 4.1, and 4.0 releases of Message Queue. For example, see New Features in Message Queue 4.5 for information about features introduced in Message Queue 4.5.

The most up-to-date version of these release notes can be found at `http://www.oracle.com/technetwork/indexes/documentation/index.html`. Check the web site prior to installing and setting up your software and then periodically thereafter to view the most up-to-date release notes and product documentation.

These release notes contain the following sections:



Third-party URLs are referenced in this document and provide additional, related information.

Oracle is not responsible for the availability of third-party Web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

# Release Notes Revision History

The following table lists the dates for all 4.x releases of the Message Queue product and describes the changes in this document associated with each release.

***Table 1–1 Revision History***

| Date | Description of Changes |
|---|---|
| February 2012 | Release of this document for Message Queue 4.5.2. |
| July 2011 | Second release of this document for Message Queue 4.5. Corrects errors and omissions, and adds information about bug 6804819. |
| February 2011 | Release of this document for Message Queue 4.5. |
| June 2010 | Release of this document for Message Queue 4.4.2. |
| June 2010 | Second release of this document for Message Queue 4.4 Update 1. Corrects errors and omissions, and adds information about bug 6925362. |
| December 2009 | Release of this document for Message Queue 4.4 Update 1. Adds new features for this release and removes outdated installation issues that applied to the previous installation program. |
| December 2009 | Second release of this document for Message Queue 4.4. Corrects errors and omissions. |
| October 2009 | Release of this document for Message Queue 4.4. Adds new features for this release. |
| May 2009 | Initial release of this document for Message Queue 4.4 Beta. Adds new features for this release. |
| December 2008 | Release of this document for Message Queue 4.3. Adds new features for this release. |
| August 2008 | Release of this document for Message Queue 4.2. Adds new features for this release. |
| September 2007 | Third release of this document for Message Queue 4.1. Adds description of support for Java Enterprise System Monitoring Framework, fixed C ports, bug fixes, and other features. |
| April 2007 | Second release of this document for Message Queue 4.1 Beta. Adds high availability feature. |
| January 2007 | Initial release of this document for Message Queue 4.1 Beta. Adds description of JAAS support. |
| May 2006 | Initial release of this document for Message Queue 4.0. |

# About Message Queue 4.5.2

Message Queue is a full-featured message service that provides reliable, asynchronous messaging in conformance with the Java Messaging Specification (JMS) 1.1. In addition, Message Queue provides features that go beyond the JMS specification to meet the needs of large-scale enterprise deployments.

Message Queue 4.5.2 is an incremental release participating in the 3.1.2 release of GlassFish Server. As a consequence, no separately downloadable, installable distribution of Message Queue 4.5.2 is available.

# Message Queue 4.5.2 Supported Platforms and Components

This section covers the following topics regarding Message Queue 4.5.2 system requirements:

## Platform Support

As a participant in the 3.1.2 release of GlassFish Server, Message Queue 4.5.2 supports the operating environments, databases, LDAP servers, and hardware listed in the *Oracle GlassFish Server 3.1 Certification Matrix*, which is accessible at the Oracle Fusion Middleware Supported System Configurations (http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html) page.

## System Virtualization Support

System virtualization is a technology that enables multiple operating system (OS) instances to execute independently on shared hardware. Functionally, software deployed to an OS hosted in a virtualized environment is generally unaware that the underlying platform has been virtualized. Oracle performs testing of its products on select system virtualization and OS combinations to help validate that Oracle products continue to function on properly sized and configured virtualized environments as they do on non-virtualized systems.

For information about Oracle support for Oracle products in virtualized environments, see Supported Virtualization Technologies with Oracle Fusion Middleware (http://www.oracle.com/technetwork/middleware/ias/overview/index.html).

## Optional Support Components

In addition to the software components listed in the *Oracle GlassFish Server 3.1 Certification Matrix*, Table 1–2 shows components that you can install to provide additional support for Message Queue clients.

***Table 1–2 Optional Support Components***

| Component | Supports | Supported Versions |
|---|---|---|
| Java Naming and Directory Interface (JNDI) | Administered object support and LDAP user repository | JNDI Version 1.2.1<br>LDAP Service Provider, Version 1.2.2<br>File System Service Provider, Version 1.2 Beta 3[1] |
| C Compiler and compatible C++ runtime library | Message Queue C clients | **Solaris:** Oracle Solaris Studio, Version 12 or later, C++ compiler with standard mode and C compiler<br>**Linux:** gcc/g++, Version 3.4.6<br>**Windows:** Microsoft Windows Visual Studio, Version 2008 SP1 |
| Netscape Portable Runtime (NSPR) | Message Queue C clients | Version 4.8.6 |
| Network Security Services (NSS) | Message Queue C clients | Version 3.12.8 |

[1] Administered object support only; supported for development and testing, but not for deployment in a production environment

# Bugs Fixed in Message Queue 4.5.2

The following table lists the bugs fixed in Message Queue 4.5.2. A bug number that starts with "MQ-" indicates an issue reported in the issue tracker of the Open Message Queue (`http://mq.java.net`) open source project.

***Table 1–3 Bugs Fixed in Message Queue 4.5.2***

| Bug | Description |
|---|---|
| 13326740 | NEWTXNLOG HOLDING EXCLUSIVE LOCK WAIT INDEFINITELY FOR A SIZE BECOMING 0 |
| 13404357 | SUBSCRIBER PRIORITY NOT MAINTAINED FOR A 2-DAEMON CLUSTER |
| MQ-116 | MQ Transaction reaper slow bottleneck seen when a lot of XA transaction stress load |
| MQ-117 | imqcmd dump trans -debug got ConcurrentModificationException |
| MQ-124 | Message delivered to consumer in a XA transaction with state STARTED at consumer client termination is appended to end of queue |
| MQ-125 | newTxnLog with imq.persist.file.sync.enabled=true, transacted remote message ack is treated as non-transacted |
| MQ-126 | newTxnLog on checkpoint holding exclusive lock wait indefinitely for playingToMessageStore.size becomes 0 |
| MQ-127 | C API (MQCRT1.DLL) dependant on non-existent MSVCR90.DLL |
| MQ-130 | The imqjmsrajar target in jarrules.xml has an error of class name. |

# Features to be Deprecated in a Future Release

The following features will be deprecated in a future release:

- **Solaris SVR4 and Linux RPM package-based distributions of the Message Queue software**

  With the release of the new multi-platform installer based on the `pkg(5)`, or IPS, system in version 4.4 Update 1, distributions of Message Queue as Solaris SVR4 packages and as Linux RPM packages are being deprecated.

- **Message-based monitoring**

  Message-based monitoring makes use of the broker's configurable Metrics Message Producer to write metrics data into JMS messages, which are then sent to metrics topic destinations, depending on the type of metrics information contained in the messages. This metrics information can then be accessed by writing a client application that subscribes to the appropriate metrics topic destination, consumes its messages, and processes the data as desired.

  The message-based monitoring feature has been supplanted by the JMX Administration API that was introduced in MQ 4.0 (see Support for JMX Administration API). The JMX API is more comprehensive (it includes more metrics data than is written to topic destinations) and is based on the JMX industry standard.

  There is no compelling reason to use message-based monitoring now that Message Queue supports the JMX API. Information about message-based monitoring will remain in the Message Queue documentation until the feature is formally deprecated.

- **Text-based installer**

The text mode of the Message Queue installer (`installer -t`) will be eliminated on all operating system platforms. In text mode, plain text is displayed in your terminal window to simulate the appearance of the graphical user interface (GUI) mode. The GUI mode and silent mode will continue to be supported.

- **Platform Support**

  Windows 2000 and Red Hat Linux 3 will no longer be supported in future releases.

- **JMSRA Resource Adapter**

  Message Queue's resource adapter, `imqjmsra.rar`, usually referred to as JMSRA, will be replaced in a future release of Message Queue by a new resource adapter. JMSRA is used to integrate Message Queue with GlassFish Server.

  The new resource adapter, which will combine the existing features of JMSRA with the features of other Oracle JMS resource adapters, will provide specialized support for Message Queue, as well as other providers, in a Java EE application server environment. As such, it will be used to integrate Message Queue into GlassFish Server and Oracle Java CAPS Suite.

# Compatibility Issues

This section covers compatibility issues regarding Message Queue 4.5.2.

## Interface Stability

Message Queue uses many interfaces that may change over time. "Stability of Message Queue Interfaces" in *Open Message Queue Administration Guide* classifies the interfaces according to their stability. The more stable an interface, the less likely it is to change in subsequent versions of the product.

## Issues Related to the Next Major Release of Message Queue

The next major release of Message Queue might introduce changes that make current Message Queue client applications incompatible with that release. This information is provided in the interest of full disclosure.

- The locations of individual files installed as part of Message Queue might change. This could break existing applications that depend on the current location of certain Message Queue files.
- Message Queue 3.5 and earlier brokers might no longer be able to operate in a cluster with newer brokers.
- In future releases, Message Queue clients might not be able to use JDK versions that are earlier than 1.5.
- In future releases, Message Queue clients might not be able to use JDK versions that are earlier than 1.6.

# Known Issues and Limitations

This section contains a list of the known issues with Message Queue 4.5.2. The following product areas are covered:

- Broker Issues
- Broker Clusters
- JMX Issues
- SOAP Support

For a list of current bugs, their status, and workarounds, Java Developer Connection members should see the Bug Parade page on the Java Developer Connection web site. Please check that page before you report a new bug. Although all Message Queue bugs are not listed, the page is a good starting place if you want to know whether a problem has been reported.

`http://bugs.sun.com/bugdatabase/index.jsp`

> **Note:** Java Developer Connection membership is free but requires registration. Details on how to become a Java Developer Connection member are provided on Sun's "For Developers" web page.

To report a new bug or submit a feature request, send mail to `imq-feedback@sun.com`.

## Installation Issues

This section describes issues related to the installation of Message Queue.

### Installing on All Platforms

These issues affect installation on all platforms.

- The Ready to Install screen displays the product name as "mq" rather than as Sun Java System Message Queue 4.3. *(Bug 6650841)*
- When the Installer is in the process of installing Message Queue 4.3 and the Progress screen is displayed, the Cancel button is active. Selecting the Cancel button at this time results in incomplete or broken installs. *(Bug 6595578)*
- The Installer Summary Screen contains a number of links that when clicked will launch a log or summary page viewer. If you dismiss this viewer window using the window close button "X" instead of the button labelled "Close', you will not be able to bring this viewer window back up. *(Bug 6587138)*

  *Workaround:* Use the button labeled Close to close the window.
- Running the installer in registration-only mode (`installer -r`) after performing a silent install in which registration was skipped, results in registration failing with "Premature end of file" error. *(Bug 6767988)*
- When running the Message Queue installer on a computer with no JDK installed, the following error message is displayed: "Invalid root in registry key "HKLM\SOFTWARE\JavaSoft\Java Runtime Environment\CurrentVersion." *(Bug 6764358)*

  *Workaround:* Install the JDK prior to running the installer.
- The mqInstallHome directory is created by the Message Queue installer even before you click the Install button on the Ready To Install screen.*(Bug 6595590)*

### Installing on Windows

When installing Message Queue on Windows, please note the following limitations.

- The installed directory structure of Message Queue 4.3 on the Windows platform is different from that of previous releases. See *Installed Directory Structure* in *Sun Java System Message Queue 4.3 Installation Guide*.
- The installer does not add entries for Message Queue to the Start>Programs menu. (*Bug 6567258*)

  *Workaround:* To start the Administration Console use the command line as shown in "Starting the Administration Console" in *Open Message Queue Administration Guide*.
- The installer does not add the `IMQ_HOME\mq\bin` directory to the `PATH` environment variable.*(Bug 6567197)*

  *Workaround:* Users need either to add this entry to their `PATH` environment variable or provide a full path name when invoking Message Queue utilities (`IMQ_HOME\mq\bin\`*command*).
- The installer does not add entries to the Windows registry to indicate that Message Queue is installed. *(Bug 6586389)*
- The installer does not add the Message Queue broker as a Windows service.

  *Workaround:* Manually add the Message Queue broker as a Windows service using the `imqsvcadm` command.
- If there is no JDK installed, the installer will throw the following error: "Invalid root in registry key HKLM\\SOFTWARE\\JavaSoft\\Java Runtime Environment\\CurrentVersion." (Bug 6764358)\par

  *Workaround:* If you see this error, install a JDK and proceed.
- When run in silent mode with an answer file, the installer returns right away. The installation does happen; but the user has no way of knowing when the silent installation is actually done.*(Bug 6586560)*
- The installer installs Message Queue on `C:\` even if the operating system is installed on a different drive. *(Bug 6673511)*
- For installation and uninstallation on Windows, there are no .bat files that the user can run, nor can user uninstall by using Add/Remove Programs in the Windows Control Panel. *(Bug 6673417)*
- On Windows Vista, you cannot install Message Queue under `C:\Program Files` unless you install from a Command Prompt as Administrator. *(Bug 6701661)*

  *Workaround:* To install from a Command Prompt as Administrator:

  1. Start>Programs>Accessories>Command Prompt.
  2. Right click on Command Prompt.
  3. Select Run as Administrator.
  4. Change directory to the Message Queue 4.2 install image.
  5. Run `installer.vbs`.
- When the uninstaller is run in dry run mode (`uninstaller -n`), it incorrectly performs an uninstall. *(Bug 6719051)*

  *Workaround:* Perform a silent install using the following command:

```
uninstaller -s
```

- The "Install Home" string on the installer Home page is not localized. *(Bug 6592491)*
- Message Queue Zip-based uninstaller hangs on Windows 2003. *(Bug 6764370)*

  *Workaround:* Manually remove the *mqInstallHome* directory.

### Installing on Solaris

- When the installer is run in dry run mode (`installer -n`), the Summary Screen shows some error messages and also displays an install status of "Incomplete". This is incorrect and misleading; a dry run does not install anything on the system; it only creates the answer file that can be subsequently used to perform a silent install. *(Bug 6594351)*
- The installer does not perform Sun Connection registration when run in silent mode with an answer file (`installer -a` *filename* `-s`). *(Bug 6710268)*

### Installing on OpenSolaris

When the installer is run in CCJK locales on OpenSolaris versions prior to 2009.06, CCJK characters display incorrectly. *(Bug 6865540)*

*Workaround:* Before running the installer, first install JDK 6 Update 12 or later.

### Installing on Linux

The following issues affect installation on the Linux Platform:

- On Red Hat Linux 5, the `compat-libstdc++` library needed to run C client applications is not included in the Message Queue distribution and is therefore is not installed by the Message Queue installer. If you are developing and running C clients, you need to install this library manually.

  The`compat-libstdc++` rpm is usually found on the install medium of the Linux version you are using. It can be installed using the following command:

  `rpm -ivh compat-libstdc++-`*x-x.x.x.x.x*`..rpm`

  where x represents the version number.

  To check that the library has been successfully installed, use the following command:

  ```
  rpm -qa | grep compat-libstdc++
  ```

- On Red Hat Linux 5, C clients can fail with a `PR_LOAD_LIBRARY_ERROR` error *(Bug 6885978)*

  On Red Hat Linux 5, C clients can fail, displaying a message similar to:

  ```
     "Preparing for NSS initialization ..."
     "Initializing NSS ..."
     "Could not connect to broker because 'PR_LOAD_LIBRARY_ERROR' (-5977)."
  producer(): Error: PR_LOAD_LIBRARY_ERROR
  ```

  This error arises because the NSS/NSPR libraries are not accessible.

  To resolve this issue, set the `LD_LIBRARY_PATH` environment variable to include the path to the NSS/NSPR libraries, `IMQ_HOME/nss/lib`.

- On the JDK Selection panel, the scroll list displays only one item. This makes it difficult to select other JDK's in the list. *(Bug 6584735)*

- If the JDK is current and the user selects "Install default JDK" on the JDK Selection Screen, the installer still tries to install it and reports that it cannot install the package. Installation completes successfully despite this issue. *(Bug 6581310)*
- If the currently installed JDK is a later version than JDK 1.5.0_15 (the version normally installed by the Message Queue installer), then the Message Queue uninstaller cannot find the default IMQ_JAVAHOME directory and returns an error. *(Bug 6673415)*

  *Workaround:* Install JDK 1.5 manually as follows before running the Message Queue uninstaller.

  ```
  # cd installImage/Product/UNIX/LINUX/X86/2.4/Packages
  ```

  ```
  # rpm -i --force jdk-1.5.0_15-linux-arch.rpm
  ```

  where *arch* is either `i586` or `amd64`.
- When the installer is run in dry run mode (`installer -n`), the Summary Screen shows some error messages and also displays an install status of "Incomplete". This is incorrect and misleading; a dry run does not install anything on the system; it only creates the answer file that can be subsequently used to perform a silent install. *(Bug 6594351)*

### Version Anomalies in the Installer

The installer displays Message Queue version information in an opaque form. *(Bug 6586507)*

**Solaris Platform** On the Solaris platform, refer to the following table to determine the Message Queue version displayed by the installer.

***Table 1–4 Version String Translation***

| Version as Displayed by the Installer on Solaris | Corresponding Message Queue Release |
|---|---|
| 4.4.1.0 | 4.4 Update 1 |
| 4.4.0.0 | 4.4 |
| 4.3.0.0 | 4.3 |
| 4.2.0.0 | 4.2 |
| 4.1.0.2 | 4.1 Patch 2 |
| 4.1.0.1 | 4.1 Patch 1 |
| 4.1.0.0 | 4.1 |
| 3.7.2.1 | 3.7 UR2 Patch 1 |
| 3.7.0.2 | 3.7 UR2 |
| 3.7.0.1 | 3.7 UR1 |
| 3.6.0.0 | 3.6 |
| 3.6.0.4 | 3.6 SP4 |
| 3.6.0.3 | 3.6 SP3 |
| 3.6.0.2 | 3.6 SP2 |
| 3.6.0.1 | 3.6 SP1 |

**Note:** For Patch releases to 3.6 SP4 (for example, 3.6 SP4 Patch 1), the releases string displayed by the installer stays the same. You need to run the command `imqbrokerd -version` to determine the exact version.

**Linux Platform** On the Linux platform, the version number displayed by the installer is in the following form.

*majorReleaseNumber.minorReleaseNumber-someNumber*

For example, 3.7-22. This tells us only that this is one of the 3.7 releases, but not which specific one. To determine the installed Message Queue version, run the command:

`imqbrokerd -version`.

### Localization Issues

The following issues relate to localization problems.

- When the installer is run in text mode (`installer -t`), in a non-English locale, multi-byte characters show up as garbage. *(Bug 6586923)*
- On the Installer Progress screen, the progress bar shows strange characters. The tooltip is hard coded in non-English locales. *(Bug 6591632)*
- The License screen of the installer displays English license text no matter which locale the Installer is run in. *(Bug 6592399)*

  *Workaround:* To access localized license files, look for at the `LICENSE_MULTILANGUAGE.pdf` file.
- Installer usage help text is not localized. *(Bug 6592493)*
- The string "None" that is seen on the Installer summary HTML page is hard coded in English. *(Bug 6593089)*
- When the installer is run in a German locale, the Welcome screen does not show the complete text that is seen in other locales. *(Bug 6592666)*
- The string "Install Home" seen on the Installer Install Home screen is not localized. It appears in English even when the installer is run in non-English locales. *(Bug 6592491)*
- When the installer is run in text mode (`installer -t`), the English response choices "Yes" and "No" are used no matter what locale the installer is run in. *(Bug 6593230)*
- The tooltip for the browse button on the Installer JDK Selection screen is hard coded in English. *(Bug 6593085)*

## Deprecated Password Option

In previous versions of Message Queue, you could use the –`p` or –`password` option to specify a password interactively for the following commands: `imqcmd`, `imqbrokerd`, and `imdbmgr`. Beginning with version 4.0, these options have been deprecated.

Instead, you can create a password file that specifies the relevant passwords and reference the password file using the `-passfile` command option, or simply enter a password when prompted by the command.

A password file can contain one or more of the passwords listed below.

- A keystore password used to open the SSL keystore. Use the `imq.keystore.password` property to specify this password.
- An LDAP repository password used to connect securely with an LDAP directory if the connection is not anonymous. Use the `imq.user_repository.ldap.password` property to specify this password.
- A JDBC database password used to connect to a JDBC-compliant database. Use the `imq.persist.jdbc.vendorName.password` property to specify this password. The *vendorName* component of the property name is a variable that specifies the database vendor. Choices include `hadb`, `derby`, `pointbase`, `oracle`, or `mysql`.
- A password to the `imqcmd` command (to perform broker administration tasks). Use the `imq.imqcmd.password` property to specify this password.

In the following example, the password to the JDBC database is set in the password file to `abracadabra`.

```
imq.persist.jdbc.mysql.password=abracadabra
```

You can use a password file in one of the following ways.

- Configure the broker to use the password file by setting the following properties in the broker's `config.properties` file.

  `imq.passfile.enabled=trueimq.passfile.dirpath=`*passwordFileDirectory*`imq.passfile.name=`*passwordFileName*

- Use the `-passfile` option of the relevant command, for example:

  `imqbrokerd -passfile` *passwordFileName*

## Administration/Configuration Issues

The following issues pertain to administration and configuration of Message Queue.

- On Windows platforms, you need to manually add the Message Queue broker as a Windows service using the `imqsvcadm` command. The installer does not do this for you.
- On Windows platforms, the built-in Windows Firewall, which is enabled by default, must be manually configured with a firewall rule that allows the broker to accept incoming connections from clients. (*Bug 6675595*)

  1. Double-click on Windows Firewall in the Control Panel

     You will have to click Continue on the User Account Control dialog for the Windows Firewall Settings dialog to open.

  2. In the Windows Firewall Settings dialog, click the Exceptions tab.
  3. Click Add program.
  4. In the Add a Program dialog, select `java.exe` and click Browse.

     Windows identifies the broker process as a Java Platform SE binary. Therefore, locate the `java.exe` used by the broker (usually at `jdk1.5.0_15\jre\bin\java.exe`).

  5. Click Change scope.
  6. In the Change Scope dialog, select "Any computer (including those on the Internet."
  7. Click OK.

8. In the Add a Program dialog, click OK.
9. In the Windows Firewall Settings dialog, click OK.

- On Windows platforms, the `imqadmin` and `imqobjmgr` commands throw an error when the `CLASSPATH` contains double quotes. (*Bug 5060769*)

  *Workaround:* Open a command prompt window and unset the `CLASSPATH`:

  ```
  set classpath=
  ```

  Then run the desired command the same command prompt window, for example:

  *mqInstallHome*`\mq\bin\imqadmin`

- The `-javahome` option in all Solaris and Windows scripts does not work if the value provided contains a space. (*Bug 4683029*)

  The `javahome` option is used by Message Queue commands and utilities to specify an alternate Java 2 compatible runtime to use. However, the path name to the alternate Java runtime must not contain spaces. The following are examples of paths that include spaces.

  Windows: `C:\jdk 1.4`

  Solaris: `/work/java 1.4`

  *Workaround:* Install the Java runtime at a location or path that does not contain spaces.

- The `imqQueueBrowserMaxMessagesPerRetrieve` attribute specifies the maximum number of messages that the client runtime retrieves at one time when browsing the contents of a queue. The attribute affects how the queued messages are batched, to be delivered to the client runtime, but it does not affect the total number of messages browsed. The attribute only affects the browsing mechanism, it does not affect queue message delivery. *(Bug 6387631)*

- On Linux platform running SELinux, the Update Center `pkg` command fails *(Bug 6892062)*

  *Workaround:* This issue is caused by a known issue in Update Center (`http://java.net/jira/browse/UPDATECENTER2-1211`). Use the following command to enable `pkg` to function on SELinux with enforcement enabled:

  ```
  # chcon -f -t textrel_shlib_t $IMAGE/pkg/vendor-packages/OpenSSL/crypto.so
  ```

## Broker Issues

The following issues affect the Message Queue broker.

- A message on a REMOVE_OLDEST destination might be marked dead after delivery to client, causing an exception when client acks the message (*Bug 6804819*)

  This situation can arise when a new message is produced to a full REMOVE_ OLDEST destination after the oldest existing message has been delivered to a client but not yet acknowledged by that client.

- Broker get an "Out of memory" error when the Queue Browser application is used (*Bug 6921483*)

  This error can arise because the messages being viewed are large or because the chunk limit for the Queue Browser (the client connection factory property

com.sun.messaging.ConnectionConfiguration.imqQueueBrowserMaxMessagesPerRetrieve) is set too high.

*Workaround:* Allocate more memory to the broker when starting it.

- Message Queue 4.4 clients receive an unclear warning when connecting to Message Queue 3.7 brokers (*Bug 6899886*)

  When a Message Queue 4.4 client connects to a Message Queue 3.7 broker, the client receives a warning of the form:

  ```
  WARNING [I500]: Caught JVM exception: ...
  [C4036]: A broker error occurred. :[505] bad version ...
  ```

  This "`bad version`" warning indicates that the client should reconnect to the broker at a lower protocol level.

- When using a JDBC data store, the database password is stored in clear text (*Bug 6691717*)

  *Workaround:* Secure the password file containing the database password as described in "Password Files" in *Open Message Queue Administration Guide*.

- Broker becomes inaccessible when persistent data store opens too many destinations. (*Bug 4953354*)

  *Workaround:* This condition is caused by the broker reaching the system open-file descriptor limit. On Solaris and Linux use the `ulimit` command to increase the file descriptor limit.

- Consumers are orphaned when a destination is destroyed. (*Bug 5060787*)

  Active consumers are orphaned when a destination is destroyed. Once the consumers have been orphaned, they will no longer receive messages (even if the destination is recreated).

- When a JMS client using the HTTP connection service terminates abruptly (for example, using `Ctrl-C`) the broker takes approximately one minute before releasing the client connection and all the associated resources.

  If another instance of the client is started within the one minute period and if it tries to use the same ClientID, durable subscription, or queue, it might receive a "Client ID is already in use" exception. This is not a real problem; it is just the side effect of the termination process described above. If the client is started after a delay of approximately one minute, everything should work fine.

- When using MySQL Cluster Edition database for a data store, the error log for MySQL Cluster Edition shows a "logging of table ./mqdb/MQCREC41Ca with BLOB attribute and no PK is not supported. *(Bug 6925362)*

  *Workaround:* Define a primary key of the CREATE_TS column for the MQCREC41 table in `default.properties`:

  ```
  imq.persist.jdbc.mysql.table.MQCREC41=\
          CREATE TABLE ${name} (\
                  RECORD MEDIUMBLOB NOT NULL,\
                  CREATED_TS BIGINT NOT NULL,\
                  PRIMARY KEY(CREATED_TS)) ${tableoption}
  ```

- When using MySQL database for a data store, storing messages greater than 1 MB throw a "Packet for query is too large..." SQLException. *(Bug 6682815)*

  *Workaround:* Start the MySQL server with the `--max_allowed_packet` option set to a value greater than the 1 MB default. For example, use the following value:

```
--max_allowed_packet=60M
```

- When using MySQL database for a highly-available shared data store, a mechanism is needed to configure the MySQL storage engine as `NDBCLUSTER`. *(Bug 6691394)*

  *Workaround:* Add the following property value to the broker's `config.properties` file (see "Enhanced Clusters: JDBC Configuration Properties" in *Open Message Queue Administration Guide*)

  ```
  imq.persist.jdbc.mysql.tableoption=EMGINE=NDBCLUSTER
  ```

- When using Oracle's 9i (JDBC 9.2.0.x) driver, broker throws "Failed to persist property..." exception. *(Bug 6626825)*

  *Workaround:* Use Oracle's 10g (JDBC 10.2.0.x) driver, for which the broker is optimized.

  ```
  imq.persist.jdbc.derby.table.MYCONSTATE41.index.IDX2=linebreakCREATE
  INDEX &(index) ON $(name) (MESSAAGE_ID)
  ```

- When using Java DB database for a data store, storing a message throws a "lock could not be obtained within the time requested" SQLException. *(Bug 6691394)*

  *Workaround:* Add the following property value to the broker's `config.properties` file::

  ```
  imq.persist.jdbc.derby.table.MYCONSTATE41.index.IDX2=linebreakCREATE
  INDEX &(index) ON $(name) (MESSAAGE_ID)
  ```

- When using IBM JVM on AIX, broker sometimes runs into low or RED memory condition without apparent reason *(Bug 6899526)*

  *Workaround:* Use the latest version of the IBM JVM (Java Runtime 1.6.0 IBM Corporation or higher) and pass the following IBM JVM GC option to `imqbrokerd`:

  ```
  # imqbrokerd -vmargs -Xgcpolicy:gencon
  ```

## Broker Clusters

The following issues affect broker clusters.

- High-availability broker with MySQL Cluster datastore fails to restart if abnormally terminated *(Bug 6896877)*

  *Workaround:* This issue is caused by a known issue in MySQL Cluster (`bugs.mysql.com/bug.php?id=47955`). A correction for this issue has been pushed to MySQL versions `5.1.39-ndb-6.3.28`, `5.1.39-ndb-7.0.9` and `5.1.39-ndb-7.1.0`.

- Only fully-connected broker clusters are supported in this release. This means that every broker in a cluster must communicate directly with every other broker in the cluster. If you are connecting brokers into a conventional cluster using the `imqbrokerd -cluster` command line argument, be careful to ensure that all brokers in the cluster are included.

- If a client is connected to a broker in an enhanced broker cluster, the client runtime will attempt to reconnect until it succeeds (it ignores the value of the `imqAddressListIterations` connection factory attribute.)

- A client can only browse the contents of queues that are located on its home broker. The client can still send messages to any queue or consume messages from any queue in the cluster; the limitation only affects queue browsing.

- In a conventional cluster that includes version 4.3 brokers, all brokers must be version 3.5 or later.
- Message Queue 4.3, 4.2, and 4.1 brokers cannot interoperate in a cluster by default with Message Queue 3.7 or 3.6 brokers because the default value of `imq.autocreate.queue.maxNumActiveConsumers` changed between these versions. *(Bug 6716400)*

  *Workaround*: Make sure all brokers have the same value of Change the value of `imq.autocreate.queue.maxNumActiveConsumers`, usually accomplished by changing the Message Queue 4.3, 4.2, and 4.1 configuration to match that used by the 3.7 or 3.6 brokers (by default, from the value of `-1` to the previous version's default value of `1`).
- To add a Message Queue 4.3 (or 4.*x*) broker to a Message Queue 3.*x* broker cluster, the a master broker must be running. *(Bug 6763796)*
- When converting from a conventional cluster to an enhanced cluster, you can use the Message Queue Database Manager utility (`imqdbmgr`) to convert an existing standalone JDBC-based data store to a shared JDBC data store as documented in "Cluster Conversion: JDBC-Based Data Store" in *Open Message Queue Administration Guide*.
- A broker using HADB cannot handle messages larger than 10 MB. *(Bug 6531734)*
- The conversion to an HADB store using the command `imqdbmgr upgrade hastore` can fail with the message "too many locks are set" if the store holds more than 10,000 message. *(Bug 6588856)*

  *Workaround*Use the following command to increase the number of locks.

  ```
  hadbm set NumberOfLocks=<desiredNumber>
  ```

  For additional information see "HADB Problems" in *Sun Java System Application Server 9.1 Enterprise Edition Troubleshooting Guide*.
- If more than 500 remote messages are committed in one transaction, the broker might return the error "HADB-E-12815: Table memory space exhausted."*(Bug 6550483)*

  For additional information, see "HADB Problems" in *Sun Java System Application Server 9.1 Enterprise Edition Troubleshooting Guide*.
- In a broker cluster, a broker will queue messages to a remote connection that has not been opened. (*Bug 4951010*)

  *Workaround:* The messages will be received by the consumer once the connection is opened. The messages will be redelivered to another consumer if the consumer's connection remains closed.
- When consuming more than one message from a remote broker in one transaction, it is possible that the following error message will be logged to the broker. The message is benign and can be ignored:

  ```
  [26/Jul/2007:13:18:27 PDT] WARNING [B2117]:
  Message acknowledgement failed from
  mq://129.145.130.95:7677/?instName=a&brokerSessionUID=3209681167602264320:
      ackStatus = NOT_FOUND(404)\
      Reason = Update remote transaction state to COMMITED(6):
  transaction 3534784765719091968 not found, the transaction
  may have already been committed.
      AckType = MSG_CONSUMED
      MessageBrokerSession = 3209681167602264320
      TransactionID = 3534784765719091968
  ```

```
        SysMessageID = 8-129.145.130.95(95:fd:93:91:ec:a0)-33220-1185481094690
        ConsumerUID = 3534784765719133952\par

[26/Jul/2007:13:18:27 PDT] WARNING Notify commit transaction
[8-129.145.130.95(95:fd:93:91:ec:a0)-33220-1185481094690,
[consumer:3534784765719133952, type=NONE]]
TUID=3534784765719091968 got response:
com.sun.messaging.jmq.jmsserver.util.BrokerException:
 Update remote transaction state to COMMITED(6):
  transaction 3534784765719091968 not found, the transaction may have already
  been committed.:
com.sun.messaging.jmq.jmsserver.util.BrokerException: Update remote transaction
state to COMMITED(6): transaction 3534784765719091968 not found, the
transaction
 may have already been committed.r
```

This message gets logged when notifying the commit to the message home broker for later messages in the transaction when the `imq.txn.reapLimit` property is low compared to the number of remote messages in one transaction. *(Bug 6585449)*

*Workaround:* To avoid this message increase the value of the `imq.txn.reapLimit` property.

### JMX Issues

On the Windows platform, the `getTransactionInfo` method of the Transaction Manager Monitor MBean returns transaction information that has incorrect transaction creation time. (*Bug 6393359*)

*Workaround:* Use the `getTransactionInfoByID` method of the Transaction Manager Monitor MBean instead.

### SOAP Support

You need to be aware of two issues related to SOAP support

- Beginning with the release of version 4.0 of Message Queue, support for SOAP administered objects is discontinued.
- SOAP development depends upon several files: `SUNWjaf`, `SUNWjmail`, `SUNWxsrt`, and `SUNWjaxp`. In version 4.1 of Message Queue, these files are available to you only if you are running Message Queue with JDK version 1.6.0 or later.
- Previously the SAAJ 1.2 implementation .jar directly referenced `mail.jar`. In SAAJ 1.3 this reference was removed; thus, Message Queue clients must explicitly put `mail.jar` in `CLASSPATH`.

## Redistributable Files

Oracle GlassFish Server Message Queue 4.4 Update 1 contains the following set of files which you may use and freely distribute in binary form:

```
fscontext.jar
imq.jar
imqjmx.jar
imqxm.jar
imqums.war
jaxm-api.jar
jms.jar
```

`libmqcrt.sl` (HP-UX)
`libmqcrt.so` (UNIX)
`mqcrt1.dll` (Windows)
In addition, you can also redistribute the `LICENSE` and `COPYRIGHT` files.

# Additional Resources

Useful Message Queue information can be found at the following Internet locations:

- Documentation

  `http://www.oracle.com/technetwork/indexes/documentation/index.html`

- Support

  `http://www.oracle.com/us/support/044752.html`

- Training

  `http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=315`

- Oracle GlassFish Server Message Queue forum

  `http://forums.oracle.com/forums/forum.jspa?forumID=874`

- Java Message Service (JMS) forum

  `http://forums.oracle.com/forums/forum.jspa?forumID=974`

# New Features in Previous Message Queue 4 Releases

The new features in previous releases of the Message Queue 4 family are described in the following sections:



## New Features in Message Queue 4.5

Message Queue 4.5 is an incremental release that includes a number of feature enhancements and bug fixes. Two of the most important features in this release relate to broker clusters, and another relates to consumer event notifications for Java clients:

#### Conventional clusters of peer brokers

This release introduces a new type of conventional cluster, the conventional cluster of peer brokers. Unlike a conventional cluster with a master broker, a conventional cluster of peer brokers maintains the cluster configuration change record in a shared JDBC data store instead of in the master broker. Thus, brokers can access cluster configuration information whether any other brokers in the cluster are running or not.

For more information about conventional clusters of peer brokers, see "Broker Clusters" in *Open Message Queue Technical Overview*. For information about configuring and managing conventional clusters of peer brokers, see "Configuring and Managing Broker Clusters" in *Open Message Queue Administration Guide*.

#### Dynamically changing the master broker

Previously, to change the master broker in a conventional cluster from one broker to another, you had to stop all brokers, manually migrate the cluster configuration change record from the old master broker to the new one, and then start all brokers. This release provides the ability to change the master broker dynamically without stopping the cluster or performing manual migration tasks. For more information, see "Changing the Master Broker in a Conventional Cluster with Master Broker" in *Open Message Queue Administration Guide*.

#### Consumer event notifications for Java clients

This release introduces consumer event notifications for Java clients, which allow a Java client to listen for the existence of consumers on a destination. Thus, for example, a producer client can start or stop producing messages to a given destination based on the existence of consumers on the destination. For more information, see "Consumer Event Notification" in *Open Message Queue Developer's Guide for Java Clients*.

## New Features in Message Queue 4.4.2

Message Queue 4.4.2 is a minor release that includes a number of feature enhancements and bug fixes. This section describes the new features included in this release.

- Message Queue now supports literal IPv6 addresses as broker host names when the *hostname:port* format is used. Previously, literal IPv6 addresses were only supported for the *hostname* format. If you use a literal IPv6 address, its format must conform to RFC2732 (`http://www.ietf.org/rfc/rfc2732.txt`), *Format for Literal IPv6 Addresses in URL's*.
- To address situations related to failover and restart of brokers in enhanced clusters, these features have been added:
  - The `-reset takeover-then-exit` option of the `imqbrokerd` command
  - The `imq.cluster.ha.takeoverWaitTimeout` broker property
- To provide more configurable control of connections to a JDBC data store, these broker properties have been added:
  - `imq.persist.jdbc.connection.timeoutIdle`
  - `imq.persist.jdbc.connection.validateOnGet`
  - `imq.persist.jdbc.connection.validationQuery`
- To control generation of informational log messages about successful message transfers across a JMS bridge, the `log-message-transfer` attribute has been added to the `jmsbridge` element in the XML configuration file for a JMS bridge.
- To enable the STOMP bridge service to bind to a specific network interface, the `imq.bridge.stomp.hostname` broker property has been added.

## New Features in Message Queue 4.4 Update 1

Message Queue 4.4 Update 1 is a minor release that includes a number of feature enhancements and bug fixes. This section describes the new features included in this release:

### New Installation Program

Message Queue 4.4 Update 1 provides a new multiplatform installer based on the `pkq(5)` system, also known as IPS or Image Packaging System. For information about this installer, see the *Sun GlassFish Message Queue 4.4 Update 1 Installation Guide*.

### Transaction Log Support for Clusters

Message Queue 4.4 Update 1 adds a transaction persistence mechanism for file-based data stores that supports broker clusters. This mechanism provides other features as well, as described in "Optimizing File-Based Transaction Persistence" in *Open Message Queue Administration Guide*.

### In-Process Broker

Message Queue 4.4 Update 1 supports running a broker from within a Java client. Such a broker, called an *in-process* or *embedded* broker, runs in the same JVM as the Java client that creates and starts it. For more information, see "Embedding a Message Queue Broker in a Java Client" in *Open Message Queue Developer's Guide for Java Clients*.

## New Features in Message Queue 4.4

Message Queue 4.4 is a minor release that includes a number of feature enhancements and bug fixes. This section describes the new features included in this release:



### JMS Bridge Service

Because the JMS specification does not define a wire protocol for communication between brokers and clients, each JMS provider (including Message Queue) has defined and uses its own propriety protocol. This situation has led to non-interoperability across JMS providers.

The JMS bridge service in Message Queue 4.4 closes this gap by enabling a Message Queue broker to map its destinations to destinations in external JMS providers. This mapping effectively allows the Message Queue broker to communicate with clients of the external JMS provider.

The JMS bridge service supports mapping destinations in external JMS providers that:

- Are JMS 1.1 compliant
- Support JNDI administrative objects
- Use connection factories of type `javax.jms.ConnectionFactory` or `javax.jms.XAConnectionFactory`
- For transacted mapping, support the XA interfaces as a resource manager

Many open source and commercial JMS providers meet these requirements, which makes the JMS bridge service an effective way to integrate Message Queue into an existing messaging environment that employs other JMS providers.

For more information about the JMS bridge service see "Configuring and Managing JMS Bridge Services" in *Open Message Queue Administration Guide*.

### STOMP Bridge Service

As mentioned earlier, the JMS specification does not define a wire protocol for communication between brokers and clients. The STOMP (Streaming Text Oriented Messaging Protocol) open source project at `http://docs.codehaus.org/display/STOMP` defines a simple wire protocol that clients written in any language can use to communicate with any messaging provider that supports the STOMP protocol.

Message Queue 4.4 provides support for the STOMP protocol through the STOMP bridge service. This service enables a Message Queue broker communicate with STOMP clients.

For more information about the STOMP bridge service see "Configuring and Managing STOMP Bridge Services" in *Open Message Queue Administration Guide*.

### Additional Enhancements

The following additional enhancements are also provided in Message Queue 4.4:

- New Universal Message Service (UMS) Functions
- IPS Package Support
- Audit Logging Feature Reinstated

**New Universal Message Service (UMS) Functions** The UMS now provides functions that use HTTP GET to offer several services:

- **getBrokerInfo**: retrieves information about the broker.
- **getConfiguration**: retrieves information about the UMS configuration.
- **debug**: turns debug logging in the UMS server on and off.
- **ping**: communicates with the broker to confirm that it is running.

For information about these new features, see "Query and utility functions using HTTP GET" in `http://mq.java.net/4.4-content/imqums/protocol.html`.

For an overview of UMS, see Universal Message Service (UMS). For documentation of the UMS API, see `http://mq.java.net/4.4-content/imqums/protocol.html`. For programming examples in several languages, see `http://mq.java.net/4.4-content/imqums/examples/README.html`.

**IPS Package Support** Message Queue is now packaged for distribution using the open source Image Packaging System (IPS), also known as the `pkg(5)` system. This packaging method has been added in order for Message Queue to integrate with Sun GlassFish Enterprise Server 2.1.1.

**Audit Logging Feature Reinstated** Message Queue 3.7 provided an audit logging feature that was removed in Message Queue 4.0. This feature has been reinstated in Message Queue 4.4. For information about this feature, see "Audit Logging with the Solaris BSM Audit Log" in *Open Message Queue Administration Guide*.

## New Features in Message Queue 4.3

Message Queue 4.3 was a minor release that included a number of feature enhancements and bug fixes. This section describes the new features included in this release:

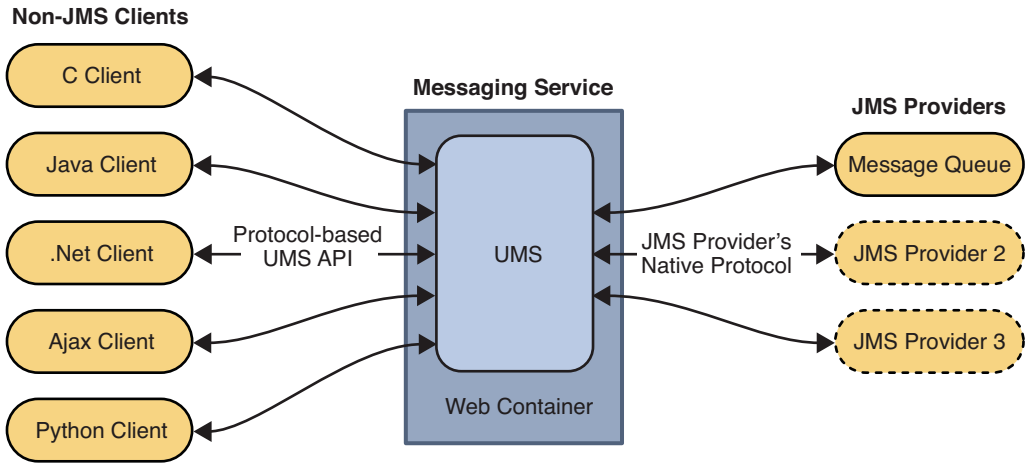
- Universal Message Service (UMS)
- AIX Platform Support
- New Zip-Based Installer
- Extended Platform Support
- Additional Enhancements

### Universal Message Service (UMS)

Message Queue 4.3 introduces a new universal messaging service (UMS) and messaging API that provides access to Message Queue from any http-enabled device. As a result, almost any application can communicate with any other application and benefit from the reliability and guaranteed delivery of JMS messaging. In addition, the UMS provides enhanced scalability for JMS messaging, allowing the number of messaging clients to reach internet-scale proportions.

**Architecture** The basic UMS architecture is shown in the following figure:

***Figure 1–1 UMS Architecture***




The UMS, which runs in a web server, is language neutral and platform independent. The UMS serves as a gateway between any non-JMS client application and a JMS provider. It receives messages sent using the UMS API, transforms them into JMS messages, and produces them as persistent messages to destinations in the JMS provider by way of the provider's native protocol. Similarly, it retrieves messages from destinations in the JMS provider in a transacted session using AUTO_ACKNOWLEDGE mode, transforms them into text or SOAP messages, and sends the messages to non-JMS clients as requested by the clients through the UMS API.

The simple, language-independent, protocol-based UMS API supports both Web-based and non-Web-based applications, and can be used with both scripting and programming languages. The API is offered in two styles: a simple messaging API that uses a Representational State Transfer (REST)-style protocol, and an XML messaging API that embeds the protocol in a SOAP message header. In both cases, however, the API requires only a single http request to send or receive a message.

The simplicity and flexibility of the UMS API means that AJAX, .NET, Python, C, Java, and many other applications can send text message and/or SOAP (with attachment) messages to JMS destinations or receive messages from JMS destinations. For example, Python applications can communicate with .NET applications, iPhone can communicate with Java applications, and so forth.

For Message Queue 4.3, the UMS supports only Message Queue as a JMS provider.

**Additional Features** The UMS serves as more than the simple gateway described above. It supports stateful as well as stateless client sessions. If requested by the client, the UMS will maintain session state for the client application across multiple service requests. The UMS can use container-managed authentication, or be configured to authenticate clients with the Message Queue broker, or both. The UMS also supports transactions, enabling client applications to commit or roll back multiple service requests as a single atomic unit.

Because the UMS can support a large number of clients on a single connection to the Message Queue broker, it eases the load on the broker's connection services, allowing for maximum scalability. In addition, UMS capacity can be increased by horizontal scaling, allowing for internet-scale messaging loads.

On the client side, because of the simplicity of the protocol-based UMS API, no client libraries are required. As a result, the API can be extended in the future to implement additional JMS features without any need to upgrade client applications.

**Using the UMS** To use the UMS, you deploy the UMS into a web container that supports Servlet 2.4 or later specifications, start the Message Queue broker, create the appropriate destinations, and write a messaging application that uses the UMS API to send or receive messages.

The UMS `imqums.war` file, contained in the Message Queue 4.3 distribution, is installed in the following location, depending on platform:

You can rename the `.war` file as appropriate.

***Table 1–5 Location of `imqums.war` file***

| Platform | Location of `imqums.war` |
|---|---|
| Solaris | `/usr/share/lib/imq` |
| Linux | `/opt/sun/mq/share/lib` |
| AIX | `IMQ_HOME/lib` |
| Windows | `IMQ_HOME\lib` |

After you have deployed the `imqums.war` into a web container at `localhost:`*port*, you can find UMS documentation at:

`http://localhost:`*port*`/imqums`

Otherwise you can find UMS documentation as follows:

- For information on configuring the UMS, see `http://mq.java.net/4.3-content/ums/config.html`.
- For documentation of the UMS API, see `http://mq.java.net/4.3-content/ums/protocol.html`.
- For programming examples in several languages, see `http://mq.java.net/4.3-content/ums/examples/README.html`.

**Supported Web Containers** UMS is currently supported on the following web containers:

- Sun GlassFish Enterprise Server, Version 2.1 and Version 3 Prelude
- Tomcat, Versions 5.5 and 6.0

### AIX Platform Support

Message Queue 4.3 provides AIX platform packages and an Installer for installing them).

The Message Queue AIX implementation supports the following software:

- AIX v 6.1 or higher (earlier versions of AIX are supported via the Unix/Java Only bundle)
- DB2 support
- IBM XL C/C++ Compiler V9.0
- JDK 1.5 or better

For installation instructions, see *AIX Installation* in *Sun Java System Message Queue 4.3 Installation Guide.*

On the AIX platform, Message Queue files are installed under a single Message Queue home directory, `IMQ_HOME`. `IMQ_HOME` denotes the directory *mqInstallHome*`/mq`, where *mqInstallHome* is the installation home directory you specify when installing the product (by default, *home-directory*`/MessageQueue`).

The resulting Message Queue directory structure is the same as that for the Windows platform (see the Windows section of "Distribution-Specific Locations of Message Queue Data" in *Open Message Queue Administration Guide*.)

Message Queue support for the AIX platform includes support for the Message Queue C-API. For instructions on building and compiling C applications on the AIX platform, see XREF.

### New Zip-Based Installer

Message Queue 4.3 introduces a new installer for Zip-based distributions, as opposed to native package distributions. The installer is used to install the new Message Queue .zip distributions for the AIX platform.

The new installer extracts Message Queue `.zip` files to any directory for which you have write access (you do not need root privileges) and it also enables you to register your Message Queue installation with Sun Connection.

To minimize the size of download bundles, the Java Runtime is no longer be included in the zip-based distribution (most sites will already have it). As a result, the `installer` command requires that a JDK or JRE be specified, either by using the `JAVA_HOME` environment variable or by using the `-j` option on the command line, as follows:

`$ installer -j` *JDK/JRE-path*

where *JDK/JRE-path* is the path of the specified JDK or JRE.

### Extended Platform Support

The following updated platform support will be certified for Message Queue 4.3:

- Oracle 11g
- Windows Server 2008

### Additional Enhancements

The following additional enhancements are included in Message Queue 4.3:

- New Directory Structure on Windows Platform
- New Broker Properties
- JMX Administration API Enhancements
- Listing Durable Subscriptions for Wildcard Subscribers

**New Directory Structure on Windows Platform** The installed directory structure for Message Queue on the Windows platform has been modified from previous versions to match that of the AIX platform. This directory structure will be adopted as well by the Solaris and Linux platforms in the future, to facilitate multiple installations on single computer and automatic update of Message Queue through Sun Connection, a Sun-hosted service that helps you track, organize, and maintain Sun hardware and software (see Installer Support for Sun Connection Registration).

**New Broker Properties** The following new properties are available for configuring a broker:

***Table 1–6 Broker Routing and Delivery Properties***

| Property | Type | Default Value | Description |
|---|---|---|---|
| `imq.transaction.producer.maxNumMsgs` | Integer | `1000` | The maximum number of messages that a producer can process in a single transaction. It is recommended that the value be less than 5000 to prevent the exhausting of resources. |
| `imq.transaction.consumer.maxNumMsgs` | Integer | `100` | The maximum number of messages that a consumer can process in a single transaction. It is recommended that the value be less than 1000 to prevent the exhausting of resources. |
| `imq.persist.jdbc.connection.limit` | Integer | `5` | The maximum number of connections that can be opened to the database. |

**JMX Administration API Enhancements** A new attribute and composite data keys have been added to the JMX API as follows:

- A NextMessageID attribute has been added to the Destination Monitor MBean to provide the JMS message ID of the next message to be delivered to a consumer.
- A NextMessageID key for composite date has been added to the Consumer Manager Monitor MBean to provide the JMS message ID of the next message to be delivered to the consumer.
- A NumMsgsPending key for composite date has been added to the Consumer Manager Monitor MBean to provide the number of messages that have been dispatched to the consumer.

For more information see "Message Queue MBean Reference" in *Open Message Queue Developer's Guide for JMX Clients*.

**Listing Durable Subscriptions for Wildcard Subscribers** The command for listing durable subscriptions:

`list dur [-d` *topicName*`]`

has been enhanced to make specification of the topic name optional. If the topic is not specified, the command lists all durable subscriptions for all topics (including those with wildcard naming conventions)

## New Features in Message Queue 4.2

Message Queue 4.2 was a minor release that included a number of new features, some feature enhancements, and bug fixes. This section describes the new features in the 4.2 release and provides further references for your use:

- Multiple Destinations for a Publisher or Subscriber
- Schema Validation of XML Payload Messages
- C-API Support for Distributed Transactions
- Installer Support for Sun Connection Registration
- Support for MySQL Database
- Additional Enhancements

For information about features introduced in Message Queue 4.1 and 4.0, see New Features in Message Queue 4.1 and New Features in Message Queue 4.0, respectively.

### Multiple Destinations for a Publisher or Subscriber

With Message Queue 4.2, a publisher can publish messages to multiple topic destinations and a subscriber can consume messages from multiple topic destinations. This capability is achieved by using a topic destination name that includes wildcard characters, representing multiple destinations. Using such symbolic names allows administrators to create additional topic destinations, as needed, consistent with the wildcard naming scheme. Publishers and subscribers automatically publish to and consume from the added destinations. (Wildcard topic subscribers are more common than publishers.)

**Note:** This feature does not apply to queue destinations.

The format of symbolic topic destination names and examples of their use is described in "Supported Topic Destination Names" in *Open Message Queue Administration Guide*.

### Schema Validation of XML Payload Messages

This feature, introduced in Message Queue 4.2, enables validation of the content of a text (not object) XML message against an XML schema at the point the message is sent to the broker. The location of the XML schema (XSD) is specified as a property of a Message Queue destination. If no XSD location is specified, the DTD declaration within the XML document is used to perform DTD validation. (XSD validation, which includes data type and value range validation, is more rigorous than DTD validation.)

For information on the use of this feature, see "Schema Validation of XML Payload Messages" in *Open Message Queue Developer's Guide for Java Clients*.

### C-API Support for Distributed Transactions

According to the X/Open distributed transaction model, support for distributed transactions relies upon a distributed transaction manager which tracks and manages operations performed by one or more resource managers. With Message Queue 4.2, the Message Queue C-API supports the XA interface (between a distributed

transaction manager and Message Queue as a XA-compliant resource manager), allowing Message Queue C-API clients running in a distributed transaction processing environment (such as BEA Tuxedo) to participate in distributed transactions.

This distributed transaction support consists of the following new C-API functions (and new parameters and error codes) used to implement the XA interface specification:

```
MQGetXAConnection()
MQCreateXASession()
```

If a C-client application is to be used in the context of a distributed transaction, then it must obtain a connection by using MQGetXAConnection() and create a session for producing and consuming messages by using MQCreateXASession(). The start, commit, and rollback, of any distributed transaction is managed through APIs provided by the distributed transaction manager.

For details of using the distributed transaction functions, see "Working With Distributed Transactions" in *Open Message Queue Developer's Guide for C Clients*.

Message Queue 4.2 provides programming examples based on the Tuxedo transaction manager. For information on the use of these sample programs, see "Distributed Transaction Sample Programs" in *Open Message Queue Developer's Guide for C Clients*.

---

**Note:** The distributed transaction functionality is supported on Solaris, Linux, and Windows platforms, however, to date it has only been certified on the Solaris platform.

---

### Installer Support for Sun Connection Registration

The Message Queue installer has been enhanced to allow for registration of Message Queue with Sun Connection, a Sun-hosted service that helps you track, organize, and maintain Sun hardware and software.

As part of Message Queue installation, you can choose to register Message Queue with Sun Connection. Information about the installed Message Queue, such as the release version, host name, operating system, installation date, and other such basic information is securely transmitted to the Sun Connection database. The Sun Connection inventory service can help you organize your Sun hardware and software, while the update service can inform you of the latest available security fixes, recommended updates, and feature enhancements.

For details of registering Message Queue with Sun Connection, see *Sun Java System Message Queue 4.3 Installation Guide*.

### Support for MySQL Database

Message Queue 4.2 introduced support for MySQL database as a JDBC-based data store. MySQL Cluster Edition can be used as a JDBC database for a standalone broker, and MySQL Cluster Edition can be used as the highly-available shared data store needed for an enhanced broker cluster. For information on configuring Message Queue to use MySQL, see "Configuring a JDBC-Based Data Store" in *Open Message Queue Administration Guide* and also "Enhanced Broker Cluster Properties" in *Open Message Queue Administration Guide*.

### Additional Enhancements

In addition to the features described above, Message Queue 4.2 included the following enhancements:

- **Remotely Produced Message Metrics**

  Message Queue 4.2 introduced new destination metrics that can be useful in monitoring destinations in a broker cluster. In a broker cluster, the messages stored in a given destination on a given broker in the cluster, consist of messages produced directly to the destination as well as messages sent to the destination from remote brokers in the cluster. In analyzing message routing and delivery in a broker cluster, it is sometimes helpful to know how many messages in a destination are local (locally produced) and how many are remote (remotely produced).

  Two new physical destination metric quantities are included in Message Queue 4.2:

  - `Num messages remote`, the current number of messages stored in memory and persistent store that were produced to a remote broker in a cluster, except for messages included in transactions.
  - `Total Message bytes remote`, the current total size in bytes of messages stored in memory and persistent store that were produced to a remote broker in a cluster, except for messages included in transactions.

  These new metric quantities are available through the `imqcmd list dst` and `imqcmd query dst` commands (see "Viewing Physical Destination Information" in *Open Message Queue Administration Guide*) and through new JMX attributes (see "Destination Monitor" in *Open Message Queue Developer's Guide for JMX Clients*).

- **Wildcard Producer and Wildcard Consumer Information**

  Information to support the use of wildcard characters in destination names (see Multiple Destinations for a Publisher or Subscriber) is provided through new monitoring data. For example, the number of wildcard producers or consumers associated with a destination are available through the `imqcmd query dst` command (see "Viewing Physical Destination Information" in *Open Message Queue Administration Guide*) and through new JMX attributes (see "Destination Monitor" in *Open Message Queue Developer's Guide for JMX Clients*). Also, wildcard information is available through the `ConsumerManager Monitor` and `ProducerManager Monitor` MBeans.

- **Support for DN Username Format for Client Authentication**

  Message Queue 4.2 introduced support for DN username format in client connection authentication against an LDAP user repository. The support involves the following new broker property (and value):

  ```
  imq.user_repository.ldap.usrformat=dn
  ```

  This property lets the broker authenticate a client user against an entry in an LDAP user repository by extracting from the DN username format the value of the attribute specified by the following property:

  ```
  imq.user_repository.ldap.uidattr
  ```

  The broker uses the value of the above attribute as the name of the user in access control operations.

  For example, if `imq.user_repository.ldap.uidattr=udi` and a client authentication username is in the format `udi=mquser,ou=People,dc=red,dc=sun,dc=com`, then "mquser" would be extracted for performing access control.

- **JAAS Authentication Enhancement**

Message Queue 4.2 introduced JAAS authentication by IP address as well as by username.

## New Features in Message Queue 4.1

Message Queue 4.1 was a minor release that included a number of new features, some feature enhancements, and bug fixes. This section describes the new features in the 4.1 release and provides further references for your use:

- High-Availability Broker Clusters
- JAAS Support
- Persistent Data Store Format Change
- Broker Environment Configuration
- Java ES Monitoring Framework Support
- Enhanced Transaction Management
- Fixed Ports for C Client Connections

For information about features introduced in Message Queue 4.0, see New Features in Message Queue 4.0.

### High-Availability Broker Clusters

Message Queue 4.1 introduced a new, enhanced broker cluster. As compared to a conventional broker cluster, which provides only *messaging service* availability (if a broker fails, another broker is available to provide messaging service), the enhanced broker cluster also provides *data* availability (if a broker fails, its persistent messages and state data are available to another broker to use to take over message delivery).

The high-availability implementation introduced in Message Queue 4.1 uses a shared JDBC-based data store: instead of each broker in a broker cluster having its own persistent data store, all brokers in the cluster share the same JDBC-compliant database. If a particular broker fails, another broker within the cluster takes over message delivery for the failed broker. In doing so, the failover broker uses data and state information in the shared data store. Messaging clients of the failed broker reconnect to the failover broker, which provides uninterrupted messaging service.

The shared JDBC-based store used in the Message Queue 4.1 high-availability implementation must itself be highly available. If you do not have a highly available database or if uninterrupted message delivery is not important to you, you can continue to use conventional clusters, which provide service availability without data availability.

To configure a Message Queue 4.1 enhanced broker cluster, you specify the following broker properties for each broker in the cluster:

- *Cluster membership properties*, which specify that the broker is in an enhanced broker cluster, the ID of the cluster, and the ID of the broker within the cluster.
- *Highly available database properties*, which specify the persistent data model (JDBC), the name of the database vendor, and vendor-specific configuration properties.
- *Failure detection and failover properties*, which specify how broker failure is detected and handled using a failover broker.

To use the enhanced broker cluster implementation, you must do the following:

1. Install a highly available database.

2. Install the JDBC driver .jar file.
3. Create the database schema for the highly available persistent data store.
4. Set high-availability properties for each broker in the cluster.
5. Start each broker in the cluster.

For a conceptual discussion of enhanced broker clusters and how they compare to conventional clusters, see "Broker Clusters" in *Open Message Queue Technical Overview*. For procedural and reference information about enhanced broker clusters, see "Configuring and Managing Broker Clusters" and "Cluster Configuration Properties" in *Open Message Queue Administration Guide*.

If you have been using a highly available database with Message Queue 4.0 and want to switch to an enhanced broker cluster, you can use the Database Manager utility (`imqdbmgr` to convert to a shared persistent data store. Also see Broker Clusters for more known issues and limitations.

### JAAS Support

In addition to the file-based and LDAP-based built-in authentication mechanisms, Message Queue 4.1 introduced support for the Java Authentication and Authorization Service (JAAS), which allows you to plug an external authentication mechanism into the broker to authenticate Message Queue clients.

For a description of the information that a broker makes available to a JAAS-compliant authentication service and an explanation of how to configure the broker to use such a service, see "Using JAAS-Based Authentication" in *Open Message Queue Administration Guide*.

### Persistent Data Store Format Change

Message Queue 4.1 changed the JDBC-based data store to support enhanced broker clusters. For this reason the format of the JDBC—based data store is increased to version 410. Format versions 350, 370, and 400 are automatically migrated to the 410 version.

Please note that the format of the file-based persistent data store remains at version 370 because no changes were made to it.

### Broker Environment Configuration

The property `IMQ_DEFAULT_EXT_JARS` has been added to the Message Queue 4.1 environment configuration file, `imqenv.conf`. You can set this property to specify the path names of external .jar files to be included in `CLASSPATH` when the broker starts up. If you use this property to specify the location of external .jar files, you no longer need to copy these files to the `lib/ext` directory. External .jar files can refer to JDBC drivers or to JAAS login modules. The following sample poperty, specifies the location of JDBC drivers.

```
IMQ_DEFAULT_EXT_JARS=/opt/SUNWhadb4/lib/hadbjdbc4.jar:/opt/SUNWjavadb/derby.jar
```

### Java ES Monitoring Framework Support

Message Queue 4.1 introduced support for the Sun Java Enterprise System (Java ES) Monitoring Framework, which allows Java ES components to be monitored using a common graphical interface. This interface is implemented by a web-based console called the Sun Java System Monitoring Console. Administrators can use the Console to view performance statistics, reate rules for automatic monitoring, and acknowledge

alarms. If you are running Message Queue along with other Java ES components, you might find it more convenient to use a single interface to manage all of them.

For information on using the Java ES monitoring framework to monitor Message Queue, see XREF.

### Enhanced Transaction Management

Previously, only transactions in a `PREPARED` state were allowed to be rolled back administratively. That is, if a session that was part of a distributed transaction did not terminate gracefully, the transaction remained in a state that could not be cleaned up by an administrator. In Message Queue 4.1, you can now use the Command utility (`imqcmd`) to clean up (roll back) transactions that are in the following states: `STARTED`, `FAILED`, `INCOMPLETE`, `COMPLETE`, and `PREPARED`.

To help you determine whether a particular transaction can be rolled back (especially when it is not in a `PREPARED` state), the Command utility provides additional data as part of the`imqcmd query txn` output: it provides the connection id for the connection that started the transaction and specifies the time when the transaction was created. Using this information, an administrator can decide whether the transaction needs to be rolled back. In general, the administrator should avoid rolling back a transaction prematurely.

### Fixed Ports for C Client Connections

In Message Queue 4.1, C clients, like Java clients, can now connect to a fixed broker port rather than to a port dynamically assigned by the broker's Port Mapper service. Fixed port connections are useful if you're trying to get through a firewall or if you need to bypass the Port Mapper service for some other reason.

To configure a fixed port connection you need to configure both the broker and the C client run time (both ends of the connection). For example, if you want to connect your client via `ssljms` to port 1756, you would do the following:

- On the client side, set the following properties:

  ```
  MQ_SERVICE_PORT_PROPERTY=1756

  MQ_CONNECTION_TYPE_PROPERTY=SSL
  ```

- On the broker side, set the `imq.`*serviceName.protocolType*`.port` property as follows:

  ```
  imq.ssljms.tls.port=1756
  ```

> **Note:** The `MQ_SERVICE_PORT_PROPERTY` connection property has been backported to Message Queue 3.7 Update 2.

## New Features in Message Queue 4.0

Message Queue 4.0 was a minor release limited to supporting Application Server 9 PE. It included a few new features, some feature enhancements, and bug fixes. This section includes a description of new features in this release:

- Support for JMX Administration API
- Client Runtime Logging
- Connection Event Notification API
- Broker Administration Enhancements

- Displaying Information About a JDBC-Based Data Store
- JDBC Provider Support
- Persistent Data Store Format Changes
- Additional Message Properties
- SSL Support

> **Caution:** One of the minor but potentially disruptive changes introduced with version 4.0 was the deprecation of the command-line option to specify a password. Henceforth, you must store all passwords in a file as described in Deprecated Password Option, or enter them when prompted.

### Support for JMX Administration API

A new API was added in Message Queue 4.0 for configuring and monitoring Message Queue brokers in conformance with the Java Management Extensions (JMX) specification. Using this API, you can configure and monitor broker functions programmatically from within a Java application. In earlier versions of Message Queue, these functions were accessible only from the command line administration utilities or the Administration Console.

For more information see the *Open Message Queue Developer's Guide for JMX Clients*.

### Client Runtime Logging

Message Queue 4.0 introduced support for client runtime logging of connection and session-related events.

Fore information regarding client runtime logging and how to configure it, see the Java Dev Guide pag 137.

### Connection Event Notification API

Message Queue 4.0 introduced an event notification API that allows the client runtime to inform an application about changes in connection state. Connection event notifications allow a Message Queue client to listen for closure and re-connection events and to take appropriate action based on the notification type and the connection state. For example, when a failover occurs and the client is reconnected to another broker, an application might want to clean up its transaction state and proceed with a new transaction.

For information about connection events and how to create an event listener, see the Java Dev Guide, page 96.

### Broker Administration Enhancements

In Message Queue 4.0, a new subcommand and several command options were added to the Command utility (`imqcmd`) to allow administrators to quiesce a broker, to shutdown a broker after a specified interval, to destroy a connection, or to set java system properties (for example, connection related properties).

- Quiescing a broker moves it into a quiet state, which allows messages to be drained before the broker is shut down or restarted. No new connections can be created to a broker that is being quiesced. To quiesce the broker, enter a command like the following.

```
imqcmd quiesce bkr -b Wolfgang:1756
```

- To shut down the broker after a specified interval, enter a command like the following. (The time interval specifies the number of seconds to wait before the broker is shut down.)

  ```
  imqcmd shutdown bkr -b Hastings:1066 -time 90
  ```

  If you specify a time interval, the broker will log a message indicating when shutdown will occur. For example,

  ```
  Shutting down the broker in 29 seconds (29996 milliseconds)
  ```

  While the broker is waiting to shut down, its behavior is affected in the following ways.

  - Administrative jms connections will continue to be accepted.
  - No new jms connections will be accepted.
  - Existing jms connections will continue to work.
  - The broker will not be able to take over for any other broker in an enhanced broker cluster.
  - The imqcmd utility will not block, it will send the request to shut down to the broker and return right away.

- To destroy a connection, enter a command like the following.

  ```
  imqcmd destroy cxn -n 2691475382197166336
  ```

  Use the command `imqcmd list cxn` or `imqcmd query cxn` to obtain the connection ID.

- To set a system property using `imqcmd`, use the new -D option. This is useful for setting or overriding JMS connection factory properties or connection-related java system properties. For example:

  ```
  imqcmd list svc -secure -DimqSSLIsHostTrusted=true
  imqcmd list svc -secure -Djavax.net.ssl.trustStore=/tmp/mytruststore
              -Djavax.net.ssl.trustStorePassword=mytrustword
  ```

For complete information about the syntax of the `imqcmd` command, see "Command Line Reference" in *Open Message Queue Administration Guide*.

### Displaying Information About a JDBC-Based Data Store

In Message Queue 4.0 a new `query` subcommand was added to the Database Manager utility, `imqdbmgr`. This subcommand is used to display information about a JDBC-based data store, including the database version, the database user, and whether the database tables have been created.

The following is an example of the information displayed by the command.

```
imqdbmgr query

[04/Oct/2005:15:30:20 PDT] Using plugged-in persistent store:
        version=400
        brokerid=Mozart1756
        database connection url=jdbc:oracle:thin:@Xhome:1521:mqdb
        database user=scott
Running in standalone mode.
Database tables have already been created.
```

### JDBC Provider Support

In Message Queue 4.0, Apache Derby Version 10.1.1 is now supported as a JDBC-based data store provider.

### Persistent Data Store Format Changes

Message Queue 4.0 introduced changes to the JDBC-based data store for optimization and to support future enhancements. For this reason the format of the JDBC-based data store was increased to version 400. Note that in Message Queue 4.0, the file-based data store version remains 370 because no changes were made to it.

### Additional Message Properties

Message Queue 4.0 added two new properties which are set on all messages that are placed in the dead message queue.

- `JMS_SUN_DMQ_PRODUCING_BROKER` indicates the broker where the message was produced.
- `JMS_SUN_DMQ_DEAD_BROKER` indicates the broker who marked the message dead.

### SSL Support

Starting with Message Queue 4.0, the default value for the client connection factory property `imqSSLIsHostTrusted` is `false`. If your application depends on the prior default value of `true`, you need to reconfigure and to set the property explicitly to `true`.

You might choose to trust the host when the broker is configured to use self-signed certificates. In this case, in addition to specifying that the connection should use an SSL-based connection service (using the imqConnectionType property), you should set the `imqSSLIsHostTrusted` property to true.

For example, to run client applications securely when the broker uses self-signed certificates, use a command like the following.

```
java -DimqConnectionType=TLS
      -DimqSSLIsHostTrusted=true ClientAppName
```

To use the Command utility (`imqcmd`) securely when the broker uses self-signed certificates, use a command like the following (for listing connector services).

```
imqcmd list svc -secure -DimqSSLIsHostTrusted=true
```

## Bugs Fixed in Previous Message Queue 4 Releases

The following sections list bugs that were fixed in their respective releases:

- Bugs Fixed in Message Queue 4.5.1
- Bugs Fixed in Message Queue 4.5
- Bugs Fixed in Message Queue 4.4.2
- Bugs Fixed in Message Queue 4.4 Update 1
- Bugs Fixed in Message Queue 4.4
- Bugs Fixed in Message Queue 4.3
- Bugs Fixed in Message Queue 4.2

## Bugs Fixed in Message Queue 4.5.1

The following table lists the bugs fixed in Message Queue 4.5.1. A bug number that starts with "MQ-" indicates an issue reported in the issue tracker of the Open Message Queue (`http://mq.java.net`) open source project.

***Table 1–7 Bugs Fixed in Message Queue 4.5.1***

| Bug | Description |
|---|---|
| 12308133 | SUNBT7019554 IMQ.JAR: MISSING IT AND PT_BR ENTRIES IN MANIFEST FILE |
| 12364646 | WARNING [B3100] INCOMPLETE(3)]: IS NOT IN STARTED(1) STATE |
| 12429252 | MAKE IMQSOCKETCONNECTTIMEOUT A PUBLIC CONNECTION FACTORY PROPERTY |
| 12584640 | BROWSER.GETENUMERATION FAILED |
| MQ-77 | exception when loading destination on ADD_CONSUMER is not propagated |
| MQ-79 | newTxnLog on Windows XP, when upgrade 4.4u2x to 4.5 broker log shows error "Could not delete ... \fs370\txn" and subquently broker unable to restart |
| MQ-80 | NoClassDefFoundError when starting JMSRA from application client downloaded using JWS |
| MQ-87 | make imqSocketConnectTimeout a public connection factory property |
| MQ-98 | example UniversalClient getUserName() needs fix |
| MQ-100 | confusing broker log message "Unexpected Broker Internal Error" when produced message expired before producer transaction commit |
| MQ-101 | confusing broker log message "Broker Internal Error" on cleanup comitted consumed message |
| MQ-102 | Topic remote message consuming is extremely slow when local consumer destination limit reached |

## Bugs Fixed in Message Queue 4.5

The following table lists the bugs fixed in Message Queue 4.5.

***Table 1–8 Bugs Fixed in Message Queue 4.5***

| Bug | Description |
|---|---|
| 6186088 | client should expire messages before delivery |
| 6761759 | broker should not expire messages that are in process delivering or have been delivered to client |
| 6788876 | Broker throws java.util.ConcurrentModificationException under heavy load |
| 6899886 | confusing log message "bad version" WARNINGs from 4.4 client runtime when connecting to 3.7 broker |
| 6925722 | If LOCAL broker can't be started due to "lock file in use", no message is reported to server log |
| 6950980 | JMX connection string (JMXServiceURL) does not use imq.jmx.hostname |
| 6961586 | msgs left in queue not delivered to any consumer after MT consumers consume 1 msg then close repeatl |

*Table 1–8 (Cont.) Bugs Fixed in Message Queue 4.5*

| Bug | Description |
|---|---|
| 6972137 | Message delivery stops when multiple consumers fetch messages from a queue (with selector) |
| 6974173 | EJB performing request/reply throws exception during commit |
| 6987430 | message expire thread repeatly reap the same message forever if the message has already removed |
| 6992678 | Message Driven Bean (MDB) does not follow remote JNDI connection factory config |
| 7011997 | C-API: disable SSL2 for NSS |

## Bugs Fixed in Message Queue 4.4.2

The following table lists the bugs fixed in Message Queue 4.4.2. Some of these issues are marked with "(OpenMQ)", which indicates the issue was reported in the issue tracker of the Open Message Queue (http://mq.java.net) open source project upon which Oracle GlassFish Server Message Queue is based.

*Table 1–9 Bugs Fixed in Message Queue 4.4.2*

| Bug | Description |
|---|---|
| 6726682 | 'imqcmd list dst' show incorrect consumer cnt after failover cause unacked msgs in cluster/w master |
| 6831953 | MQ JDBC connection pool fails to recover from DB outage if driver is type ConnectionPoolDataSource |
| 6914395 | Broker stops responding if the message in the Topic of durable subscriber is modified by admin event |
| 6918792 | Incorrect synchronization in IMQDirectService |
| 6931209 | Confusing INFO messages logged during JMS application initialisation |
| 6932420 | On store message, JDBC conn.rollback() is not called to rollback the db txn when exception |
| 6937110 | MQAddress utility class url parsing does not work if IPv6 address |
| 6943892 | imqbrokerd uses private SUN JDK flag unsupported by BEA JRockit VM |
| 6947108 | Spurious message when RA EndpointConsumer created |
| 42 (OpenMQ) | Incorrect synchronization in IMQDirectService |

## Bugs Fixed in Message Queue 4.4 Update 1

The following table describes the bugs fixed in Message Queue 4.4 Update 1. Some of these issues are marked with "(OpenMQ)", which indicates the issue was fixed in the Open Message Queue (http://mq.java.net) open source project upon which Oracle GlassFish Server Message Queue is based.

*Table 1–10 Bugs Fixed in Message Queue 4.4 Update 1*

| Bug | Description |
|---|---|
| 6590909 | DIRECT mode MDB does not connect to remote broker when addresslist is overridden |
| 6616704 | Broker memory growth when many consumers created within a Session |

*Table 1–10 (Cont.) Bugs Fixed in Message Queue 4.4 Update 1*

| Bug | Description |
|---|---|
| 6745761 | XAResource.isSameRM() should return true when two connections used in same XA TX (with JMSJCA) |
| 6745763 | XAResource.isSameRM() should return true when two connections used in same XA TX (JMSRA DIRECT mode) |
| 6745768 | XAResource.isSameRM() should return true when two connection used in same XA TX (JMSRA LOCAL/REMOTE) |
| 6760450 | Message store getting corrupted if the machine is rebooted without stopping the MQ (GF) instance |
| 6766241 | UMS: SendMsg.html AJAX example uses /ums as default context root. It should use /imqums |
| 6766852 | DirectXAResource translates broker CONFLICT status to "TxID is already in use" |
| 6799428 | Non-persistent messages/Non-durable deposited messages in DMQ cannot be consumed but browsable. |
| 6799428 | Non-persistent messages/Non-durable deposited messages in DMQ cannot be consumed but browsable. |
| 6809353 | openmq 4.3 HA with posgtresql (8.1) doesn't work (imqbrokerd can't start ) |
| 6809750 | Connection pooling (from JMSRA) for clientId connection does not work. |
| 6812198 | Classcast exception thrown when monitoring using MQ topic metrics |
| 6832000 | MQ reapExcessConnection JDBC connection runs into HIGH CPU spin |
| 6833109 | MQClusterMonitor JMX Sample application throws Exception on AIX with JDK6 |
| 6835420 | Default value of NoGCDefault calculated incorrectly. May cause excessive GC when memory is low. |
| 6852018 | Error message "Cannot add durable consumer {0}. No ClientID was set on connection." is misleading |
| 6856991 | NullPointerException after broker restarts then rollback a durable consumer PREPARED transaction |
| 6874125 | WARNING: MQJMSRA_DC2001: connectionId=555670328604044289:_ destroy():called on a connection... |
| 6878945 | RFE: JMSBridge: allow specify username/password to create connection from connection-factory |
| 6881493 | Admin temporary destinations should not be stored for HA broker |
| 6881753 | RFE JMSBridge: allow tag each message with the jmsbridge name before transfer to target |
| 6884673 | MQ 4.4 Broker could not establish cluster connection with MQ 3.7/3.6 Broker |
| 6886390 | Persist/Txn published msgs went to DMQ can cause mq.sys.dmq not found err when consume them from DMQ |
| 6886515 | AccessControlException when using JMX to delete a destination in an embedded broker |
| 6890628 | setting the broker property "imq.autocreate.destination.isLocalOnly=true" has no effect |
| 6891615 | Selector does not always work when running broker 4.3 in glassfish |
| 6891624 | Msgs 'Remote' can become higher than 'Count' in 'imqcmd list dst' |

*Table 1–10 (Cont.) Bugs Fixed in Message Queue 4.4 Update 1*

| Bug | Description |
|---|---|
| 6891629 | need user-friendly message when arithmetic exeception occurs in selector |
| 6891717 | ifimq.transaction.autorollback=true,autorollback PREPARED ack not clear cause TransactionAckExistEx |
| 6891802 | "[B4061]:Can not use Transaction ID..currently in use"on broker restart after takeover remote tx ack |
| 6892512 | Memory Leak: Temporary Destinations are not removed from connection when tempDest.delete() is called |
| 6895040 | if masterbroker has temp dest,slave broker fail get uidprefix on start after uidprefix lock timeouts |
| 6896230 | new consumer created on masterb while masterb restart after sync/w slaves maynot propagate to all |
| 6896764 | equals method on TransactionAcknowledgement is incorrect. |
| 6898355 | takeover lock reseted in cluster managr init on broker restart without waiting for takeover complete |
| 6901405 | RFE: log JDBC vendor information and vendor properties if specified |
| 16 (OpenMQ) | Selector does not always work when running broker 4.3 in glassfish |
| 17 (OpenMQ) | openmq 4.3 HA with posgtresql (8.1) doesn't work (imqbrokerd can't start ) |
| 22 (OpenMQ) | installer references non existing binary and fails |
| 25 (OpenMQ) | Memory leak when creating TemporaryTopic. |
| 29 (OpenMQ) | Broker Isolation |
| 30 (OpenMQ) | Msgs 'Remote' can become higher than 'Count' in 'imqcmd list dst' |
| 31 (OpenMQ) | need user-friendly message when arithmetic exc. occurs in selector |
| 32 (OpenMQ) | fix for int> long overflows |
| 33 (OpenMQ) | OpenMQ installer: "Invalid SwiXML Descriptor" error when ran under ja locale |

## Bugs Fixed in Message Queue 4.4

The following table describes the bugs fixed in Message Queue 4.4.

*Table 1–11 Bugs Fixed in Message Queue 4.4*

| Bug | Description |
|---|---|
| 6242247 | MQ cluster with masterbroker startup and hangs if both broker is on same machine have same name |
| 6760937 | Broker does not reconnect to the DB if it is restarted |
| 6763252 | broker should log a meaningful message than NPE when ack a message that has been expired/removed |
| 6765410 | masterbroker sends local interests 2 times cause slave exception Durable subscription already active |
| 6796506 | remote PREPARED msg not redelivered after rollback in case timeout in receiving remote PREPARE reply |
| 6807708 | TemporaryDestination.delete fail if master broker is not running |
| 6812037 | RFE: pass MQ_CALLBACK_RUNTIME_ERROR to afterMessageDelivery if MQMessageListenerFunc returns error |

*Table 1–11 (Cont.) Bugs Fixed in Message Queue 4.4*

| Bug | Description |
|---|---|
| 6812755 | FINE level log message should be WARNING if before/afterMessageDelivery callbacks return error |
| 6816023 | Message.setStringProperty() exception does not show property name on Illegal character exception |
| 6819095 | RFE: cluster should support setting input/output stream buffer size and TcpNoDelay |
| 6820585 | 'imqcmd list txn' does not show COMMITTED cluster transactions waiting for remote broker completion |
| 6820588 | a cluster transaction that consume both local and remote messages stay as COMMITTED in waiting state |
| 6821639 | NPE on rollback/commitTransaction during AS recovery for MQRA-DIRECT mode |
| 6823364 | RFE: upgrade C-API compiler to Sun Studio 12 on Solaris |
| 6829113 | ConcurrentModificationException when Tuxedo TM rollback timed out transaction under heavy load |
| 6832197 | non-transacted remote ack should not wait for remote reply if client does not ask for ackack |
| 6834735 | confusing log msg "Unexpected Broker Interal Error" when Tuxedo TM timeout a txn in START state |
| 6836364 | wildcard subscriber does not receive remote msg if its topic is created before subscriber |
| 6836691 | HA(JCAPS):msg already been removed exception on receive after XA receiver rollback then commit a msg |
| 6836749 | HA(JCAPS):ack exists in store exception on receive after 1 of durables rollback then commit a msg |
| 6837671 | HA(JCAPS):endless redeliver a committed message when XAResourceImpl.rollback after a success commit |
| 6839193 | RFE: upgrade C++ compiler to Visual Studio 2008 SP1 |
| 6845625 | broker entering low memory state when remote consumers repeatly created/closed |
| 6852207 | NPE on sending msg to remote broker causes remote broker "unable to process message" on read msg pkt |
| 6853822 | confusing exception message "Cannot perform operation END_TRANSACTION" when end a FAILED txn |
| 6854142 | "Waiting for cluster connection" "Closed cluster connection" to remote broker every 3 minutes |
| 6858121 | confusing WARNING 'Unknow transaction' in broker log on 'imqcmd list txn' if remote txn exists |
| 6858488 | COMMITTED txn not removed from txn home broker if remote participant broker removed its COMMITTED tx |
| 6858905 | ConcurrentModificationException in Consumer.destroyConsumer |
| 6861362 | RFE: JMSBridge: support auto-map target destination to source Message.getJMSDestination |
| 6861528 | RFE: JMSBridge: allow MessageTransformer.transform() branch msg to a different destination in target |
| 6861653 | excessive cluster txn info sent to COMMIT incomplete down remote broker under high txn load |

*Table 1–11 (Cont.) Bugs Fixed in Message Queue 4.4*

| Bug | Description |
|---|---|
| 6862413 | confusing log message "mq://xxx.xxx.xx.xx:pppp/ ..." is reachable within 60 seconds" |
| 6863867 | MissingResourceException on HA broker restarts if has pending COMMITTED from a down remote broker |
| 6867596 | recovered PREPARED txn after broker restart return backto PREPARED state if broker restart again |
| 6868525 | NullPointerException on forwarding temporary destination to remote broker on link establishment |
| 6868578 | some broadcast/unicast no check if a link established interferes/w link handshake cause link down |
| 6871612 | HA:log msgs"Cant notify transaction.completion.."when consume remote msgs if the pending broker down |
| 6886391 | NullPointerException on acknowledge message if message has been removed already |

## Bugs Fixed in Message Queue 4.3

The following table describes the bugs fixed in Message Queue 4.3.

*Table 1–12 Bugs Fixed in Message Queue 4. 3*

| Bug | Description |
|---|---|
| 6634033 | Cluster protocol does not propagate value of `imqConsumerFlowLimit` to remote brokers when a client is created. |
| 6713012 | Destruction of a consumer on a broker in a cluster at the same time that a remote broker is being restarted can result in some messages not being delivered. |
| 6727555 | Broker log message "Max bytes per msg exceeded" has the actual message size and the `max bytes per message` values switched. |
| 6737404 | JMX metrics need to provide counts of messages dispatched from destinations (topics and queues) but yet to be delivered to consumers. |
| 6740568 | Broker throws an exception when consuming too many messages in a single transaction. |
| 6758524 | The command to list durable subscriptions (`imqcmd list dur -d "foo.*"`) does not accept wildcard characters in the destination name. |
| 6758952 | Setting `imq.portmapper.hostname=localhost` causes brokers to be unable to connect into a cluster. |
| 6758817 | Setting `imq.cluster.hostname=localhost` (not recommended) causes brokers on different machines to be unable to connect into a cluster. |

## Bugs Fixed in Message Queue 4.2

The following table describes the bugs fixed in Message Queue 4.2.

*Table 1–13 Bugs Fixed in Message Queue 4.2*

| Bug | Description |
|---|---|
| 6581592 | When the installer or uninstaller is run in text mode (`installer -t`), the Summary screen shows the directory containing the log/summary files but does not list the names of these files. |
| 6585911 | The installer's JDK Selection screen incorrectly includes the JRE bundled with the installer and used to run the installer. |

*Table 1–13 (Cont.) Bugs Fixed in Message Queue 4.2*

| Bug | Description |
|---|---|
| 6587112 | The installer summary screen shows garbage in multi-byte locales. |
| 6587127 | When running the installer by referencing an answer file (`installer -a` *filename* `-s`), if the answer file does not exist, the error messages are inconsistent and unclear. |
| 6590969 | Allows DN username format in client connection authentication. |
| 6594381 | Installation of Message Queue 4.1 localization RPM's (which happens when you select the "Install Message Queue multilingual packages" checkbox on the Multilingual Packages screen) will fail if older versions of Message Queue localization RPM's exist on your system. |
| 6599144 | When uninstalling Message Queue 4.2, splash screen and uninstaller hangs and screens appear empty and gray on Java SE 6, but work on Java SE 5. |
| 6615741 | Message delivered in a transacted consumer session that is rolled back is not redelivered if the original consumer closed before rollback. |
| 6629922 | Distributed transaction handler does not redeliver message to inactive consumer in correct order. |
| 6635130 | Broker fails to notify producer of non persistent messages to resume production after having been paused because destination had reached memory or message limits. |
| 6641117 | Message delivered in a transacted consumer session that is rolled back is not redelivered if the original consumer closed after rollback. |
| 6683897 | Message Queue installer's summary screen reports configuration error even though configuration appears to complete successfully: installer cannot write to `/dev/sterr` on some computers. |
| 6684069 | In broker cluster in which large number of messages are delivered to remote client in consumer transaction, commit transaction fails. |
| 6688935 | Default value of Portmapper read timeout is too small. |
| 6695238 | C-client applications cannot connect to a broker installed in a location that has spaces in the path. |
| 6710168 | Consumer no longer consumes messages if destination is paused twice without being resumed between the pauses. |
| 6710169 | JMX operation `ConsumerManagerMonitor.getConsumerInfo` always returns `SESSION_TRANSACTED` for the acknowledgement mode. |

## Bugs Fixed in Message Queue 4.1

The following table describes the bugs fixed in Message Queue 4.1.

*Table 1–14 Bugs Fixed in Message Queue 4.1*

| Bug | Description |
|---|---|
| 6381703 | Transacted remote messages can be committed twice if the broker originating the message restarts. |
| 6388049 | Cannot clean up an uncompleted distributed transaction. |
| 6401169 | The commit and rollback options for imqcmd do not prompt for confirmation. |
| 6473052 | Default for autocreated queues should be round robin. (`MaxNumberConsumers` = -1). |
| 6474990 | Broker log shows `ConcurrentModificationException` for`imqcmd list dst` command. |
| 6487413 | Memory leak when limit behavior is `REMOVE_OLDEST` or `REMOVE_LOWER_PRIORITY`. |
| 6488340 | Broker spins, and client waits for reply to acknowledge. |

*Table 1–14 (Cont.) Bugs Fixed in Message Queue 4.1*

| Bug | Description |
|---|---|
| 6502744 | Broker does not honor the dead message queue's default limit of 1000 messages. |
| 6517341 | Client runtime needs to improve reconnect logic when the client is connected to an enhanced broker cluster by allowing the client to reconnect no matter what the value of the *imqReconnectEnabled* property is. |
| 6528736 | Windows automatic startup service (*imqbrokersvc*) crashes during startup. |
| 6561494 | Messages are delivered to the wrong consumer when both share a session. |
| 6567439 | Produced messages in a `PREPARED` transaction are delivered out of order if they are committed after broker restarts. |

## Bugs Fixed in Message Queue 4.0

The following table describes the bugs fixed in Message Queue 4.0.

*Table 1–15 Bugs Fixed in Message Queue 4.0*

| Bug Number | Description |
|---|---|
| 4986481 | In Message Queue 3.5, calling `Session.recover` could hang in auto-reconnect mode. |
| 4987325 | Redelivered flag was set to `false` for redelivered messages after calling `Session.recover`. |
| 6157073 | Change new connection message to include the number of connections on the service in addition to the total number of connections. |
| 6193884 | Message Queue outputs garbage message to syslog in locales that use non-ASCII characters for messages. |
| 6196233 | Message selection using `JMSMessageID` doesn't work. |
| 6251450 | `ConcurrentModificationException` on `connectList` during cluster shutdown. |
| 6252763 | `java.nio.BufferOverflowException` in `java.nio.HeapByteBuffer.putLong/Int`. |
| 6260076 | First message published after startup is slow with Oracle storage. |
| 6260814 | Selector processing on `JMSXUserID` always evaluates to `false`. |
| 6264003 | The queue browser shows messages that are part of transactions that have not been committed. |
| 6271876 | Connection Flow Control does not work properly when closing a consumer with unconsumed messages. |
| 6279833 | Message Queue should not allow two brokers to use the same jdbc tables. |
| 6293053 | Master broker does not start up correctly if the system's IP address is changed, unless the store is cleared (using `–reset store`.) |
| 6294767 | Message Queue broker needs to set `SO_REUSEADDR` on the network sockets it opens. |
| 6304949 | Unable to set `ClientID` property for `TopicConnectionFactory`. |
| 6307056 | The txn log is a performance bottleneck. |
| 6320138 | Message Queue C API lacks ability to determine the name of a queue from a reply-to header. |
| 6320325 | The broker sometimes picks up JDK 1.4 before JDK 1.5 on Solaris even if both versions are installed. |

*Table 1–15 (Cont.) Bugs Fixed in Message Queue 4.0*

| Bug Number | Description |
|---|---|
| 6321117 | Multibroker cluster initialization throws `java.lang.NullPointerException`. |
| 6330053 | The jms client throws `java.lang.NoClassDefFoundError` when committing a transaction from the subscriber. |
| 6340250 | Support `MESSAGE` type in C-API. |
| 6351293 | Add Support for Apache Derby database. |