Sun Java System Message Queue 4.3 Release Notes

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–6360
December, 2008

081210@21288

Contents

◆ ◆ ◆ CHAPTER 1

Sun Java System Message Queue 4.3 Release Notes

Version 4.3

Part Number 820-6361

These release notes contain important information available at the time of release of Sun Java™ System Message Queue 4.3. New features and enhancements, known issues and limitations, and other information are addressed here. Read this document before you begin using Message Queue 4.3.

These release notes also contain information about the 4.2, 4.1, and 4.0 releases of Message Queue. For example, see "New Features in Message Queue 4.2" on page 16, "New Features in Message Queue 4.1" on page 19, and "New Features in Message Queue 4.0" on page 22, respectively, for information about features introduced in those releases.

The most up-to-date version of these release notes can be found at the Sun Java System Message Queue documentation web site, `http://docs.sun.com/coll/1307.6`. 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:

- "Additional Sun Resources" on page 46

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

Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun 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. Sun 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
May 2006	Initial release of this document for Message Queue 4.0.
January 2007	Initial release of this document for Message Queue 4.1 Beta. Adds description of JAAS support.
April 2007	Second release of this document for Message Queue 4.1 Beta. Adds high availability feature.
September 2007	Third release of this document forMessage Queue 4.1. Adds description of support for Java Enterprise System Monitoring Framework, fixed C ports, bug fixes, and other features.
August 2008	Release of this document forMessage Queue 4.2. Adds new features for this release.
December 2008	Release of this document forMessage Queue 4.3. Adds new features for this release.

Installing or Upgrading to Message Queue 4.3

You can perform a fresh install of Message Queue 4.3 or an upgrade from Message Queue 3.6 (or later) by using the Message Queue 4.3 installer. The procedure and all other information relevant to installing or upgrading on the Solaris, Linux, AIX, and Windows platforms is documented in the *Sun Java System Message Queue 4.3 Installation Guide*, which has been updated for Message Queue 4.3.

If you are upgrading from a version of Message Queue earlier than version 3.6, refer to the *Sun Java Enterprise System 5 Upgrade Guide for UNIX*, *Sun Java Enterprise System 5 Update 1 Upgrade Guide for UNIX*

Also, please check "Installation Issues" on page 33 for known installation and upgrade issues and limitations.

Message Queue 4.3 Supported Platforms and Components

This section covers the following topics regarding Message Queue 4.3 system requirements:

Operating System Platform Support

Message Queue 4.3 is supported on Solaris, Linux, Windows, and AIX operating system platforms. Table 1–2 shows the supported versions of each of these platforms. For the hardware requirements of each platform see the *Sun Java System Message Queue 4.3 Installation Guide*

TABLE 1–2 Supported Platform Versions

Platform	Supported Versions
Solaris	Solaris 9 (SunOS 5.9), all updates (SPARC, *x*86) Solaris 10 (SunOS 5.10), all updates (SPARC, *x*86, *x*64)
Linux	Red Hat Enterprise Linux Advanced Server 3.0, 4.0, 5.0, all updates, 32– and 64–bit versions (*x*86, *x*64) Red Hat Enterprise Linux Enterprise Server 3.0, 4.0, 5.0, all updates, 32– and 64–bit versions (*x*86, *x*64)
AIX	AIX 6.1

TABLE 1–2 Supported Platform Versions *(Continued)*

Platform	Supported Versions
Windows	Windows Vista
	Windows XP Professional, SP2 (*x*86)[1]
	Windows 2000 Advanced Server, SP4 (*x*86)[2]
	Windows Server 2003 Standard and Enterprise Editions, SP2, 32– and 64–bit versions (*x*86, *x*64)[3]
	Windows Server 2008 Standard and Enterprise Editions, SP2, 32– and 64–bit versions (*x*86, *x*64)[3]

[1] No Home, Tablet PC, or Media Center Edition support

[2] No Professional or Server Edition support

[3] No Web or Small Business Server Edition support

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. Sun performs testing of its Sun Java System products on select system virtualization and OS combinations to help validate that the Sun Java System products continue to function on properly sized and configured virtualized environments as they do on non-virtualized systems. For information about Sun support for Sun Java System products in virtualized environments, see http://docs.sun.com/doc/820-4651.

Component Dependencies

In addition to platform-specific requirements, Message Queue 4.3 also depends on certain basic components that must be installed in order to develop and run Message Queue clients. Table 1–3 describes these components. Other versions or vendor implementations can also be used, but they are untested by Sun Microsystems and therefore not officially supported.

Note – The Message Queue Installer for the Solaris, Linux, and Windows platforms, allows you to select an existing JDK/JRE or to install the JDK version (1.5.0_15).

TABLE 1–3 Required Support Components

Component	Supports	Supported Versions[1]
Java Runtime Environment (JRE)	Message Queue broker and administration tools	J2SE™ Runtime Environment 1.5.0_15 or later Java™ SE Runtime Environment 1.6.0_10
Java Software Development Kit (JDK), Standard Edition	Java client development and deployment	J2SE Development Kit 1.5.0_15 or later Java SE Development Kit 1.6.0_10

[1] Sun Microsystems production versions only

Table 1–4 shows additional components that you can install to provide additional support for Message Queue clients. You may not need all of the components listed: for example, if you are not writing a C client, you will not need the C compiler, C++ runtime library, NSPR, or NSS.

TABLE 1–4 Optional Support Components

Component	Supports	Supported Versions
Application server	HTTP/HTTPS	Sun Java System Application Server Enterprise Edition, Version 9.1 .1 (GlassFish Enterprise Server 2.1)
Web server	HTTP/HTTPS	Sun Java System Web Server Enterprise Edition, Version 7.0, Update 3
Database	JDBC-based data store	HADB, Version 4.4.3–6 Java DB (Apache Derby), Version 10.4 MySQL Community/Enterprise Edition, Version 5.0 Oracle 9i, 10g, and 11g postgreSQL, Version 8.1 **Note –** The PointBase database is no longer supported.
Highly-available database	High-availability broker clusters	HADB, Version 4.4.3–6 MySQL Cluster Edition, Version 5.0 Oracle10g and 11g
Lightweight Directory Access Protocol (LDAP) directory server	Message Queue user repository and administered objects	Sun Java System Directory Server, Version 6.0

TABLE 1–4 Optional Support Components *(Continued)*

Component	Supports	Supported Versions
Java Naming and Directory Interface (JNDI)	Administered object support and LDAP user repository	JNDI Version 1.2.1 LDAP Service Provider, Version 1.2.2 File System Service Provider, Version 1.2 Beta 3[1]
C Compiler and compatible C++ runtime library	Message Queue C clients	**Solaris:** Sun Studio, Version 11 or later, C++ compiler with standard mode and C compiler **Linux:** gcc/g++, Version 3.2.3 **Windows:** Microsoft Windows Visual C++, Version 6.0 SP3
Netscape Portable Runtime (NSPR)	Message Queue C clients	Version 4.7[2]
Network Security Services (NSS)	Message Queue C clients	Version 3.11.9[2]

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

[2] Bundled as a shared package in the download bundle

New Features in Message Queue 4.3 and Recent Releases

The new features in Message Queue 4.3 and previous releases in the Message Queue 4.*x* family are described in the following sections:

New Features in Message Queue 4.3

Sun Java System 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 4.3 is a minor release that includes a number of feature enhancements and bug fixes. This section describes the new features included in this release:

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:

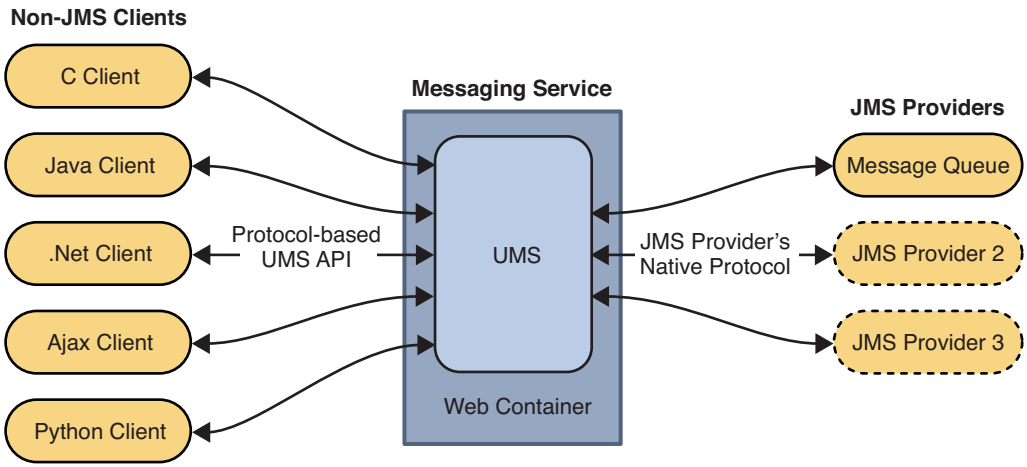

FIGURE 1–1 UMS Architecture

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 to destinations in the JMS provider by way of the provider's native protocol. Similarly, it retrieves messages from destinations in the JMS provider, 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`

TABLE 1–5 Location of `imqums.war` file *(Continued)*

Platform	Location of `imqums.war`
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 `https://mq.dev.java.net/4.3-content/ums/config.html`.
- For documentation of the UMS API, see `https://mq.dev.java.net/4.3-content/ums/protocol.html`.
- For programming examples in several languages, see `https://mq.dev.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 Chapter 4, "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 Appendix A, "Platform-Specific Locations of Message Queue Data," in *Sun Java System Message Queue 4.3 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" on page 14
- "New Broker Properties" on page 15
- "JMX Administration API Enhancements" on page 15
- "Listing Durable Subscriptions for Wildcard Subscribers" on page 15

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" on page 17).

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 Chapter 3, "Message Queue MBean Reference," in *Sun Java System Message Queue 4.3 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:

For information about features introduced in Message Queue 4.1 and 4.0, see "New Features in Message Queue 4.1" on page 19 and "New Features in Message Queue 4.0" on page 22, 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 *Sun Java System Message Queue 4.3 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" on page 16.

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 *Sun Java System Message Queue 4.3 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 *Sun Java System Message Queue 4.3 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 *Sun Java System Message Queue 4.3 Administration Guide* and also "High-Availability Cluster Properties" in *Sun Java System Message Queue 4.3 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` and `Total Message bytes remote`. The new metric quantities are available through the `imqcmd list dst` and `imqcmd query dst` commands (see "Viewing Physical Destination Information" in *Sun Java System Message Queue 4.3 Administration Guide*) and through new JMX attributes (see "Destination Monitor" in *Sun Java System Message Queue 4.3 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" on page 16) 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 *Sun Java System Message Queue 4.3 Administration Guide*) and through new JMX attributes (see "Destination Monitor" in *Sun Java System Message Queue 4.3 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" on page 19
- "JAAS Support" on page 20
- "Persistent Data Store Format Change" on page 21
- "Broker Environment Configuration" on page 21
- "Java ES Monitoring Framework Support" on page 21
- "Enhanced Transaction Management" on page 21
- "Fixed Ports for C Client Connections" on page 22

For information about features introduced in Message Queue 4.0, see "New Features in Message Queue 4.0" on page 22.

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 Chapter 4, "Broker Clusters," in *Sun Java System Message Queue 4.3 Technical Overview*. For procedural and reference information about enhanced broker clusters, see Chapter 10, "Configuring and Managing Broker Clusters," in *Sun Java System Message Queue 4.3 Administration Guide* and "Cluster Configuration Properties" in *Sun Java System Message Queue 4.3 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" on page 42 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 *Sun Java System Message Queue 4.3 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:

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" on page 39, 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 *Sun Java System Message Queue 4.3 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 Chapter 15, "Command Line Reference," in *Sun Java System Message Queue 4.3 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
```

Features to be Deprecated in a Future Release

The following features will be deprecated in a future release:

- **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" on page 23). 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 the Sun Java System Application Server.

  The new resource adapter, which will combine the existing features of JMSRA with the features of other Sun JMS resource adapters, will provide specialized support for Message Queue, as well as other providers, in a Java EE 5 Application Server environment. As such, it will be used to integrate Message Queue into Sun GlassFish Enterprise Server and Sun Java Composite Application Platform Suite (Java CAPS).

Bugs Fixed in Message Queue 4.3 and Recent Releases

Message Queue 4.3 includes new bug fixes and also incorporates bugs that were fixed in the previous releases in the Message Queue 4.*x* family.

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

Bugs Fixed in Message Queue 4.3

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

TABLE 1–7 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 1–8 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.
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 1–9 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.
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 1–10 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.

TABLE 1–10 Bugs Fixed in Message Queue 4.0 *(Continued)*

Bug Number	Description
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.
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.

Documentation Updates in Message Queue 4.3

This section contains information regarding Message Queue 4.2 documentation updates:

Compatibility Issues

This section covers compatibility issues regarding Message Queue 4.3.

Interface Stability

Sun Java System Message Queue uses many interfaces that may change over time. Appendix B, "Stability of Message Queue Interfaces," in *Sun Java System Message Queue 4.3 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 appliations incompatible with that release. This information is provided in the interest of full disclosure.

- The locations of individual files installed as part of Sun Java System 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.

Changes in the Message Queue 4.3 Documentation Set

The Message Queue 4.3 documentation set includes updates to the Message Queue 4.2 documentation set as described below:

Technical Overview

The *Sun Java System Message Queue 4.3 Technical Overview* reflects new features in Message Queue 4.3.

Installation and Upgrade Information

The *Sun Java System Message Queue 4.3 Installation Guide* includes installation of Message Queue on the AIX platform.

Administration Guide

The *Sun Java System Message Queue 4.3 Administration Guide* includes minor bug fixes, support for the AIX platform, and revised procedures for managing broker clusters and converting conventional clusters to enhanced clusters.

Developer's Guide for Java Clients

The *Sun Java System Message Queue 4.3 Developer's Guide for Java Clients* reflects new version numbers, but otherwise has not been revised.

Developer's Guide for C Clients

The *Sun Java System Message Queue 4.3 Developer's Guide for C Clients* includes information on building C client applications on the AIX platform.

Developer's Guide for JMX Clients

The *Sun Java System Message Queue 4.3 Developer's Guide for JMX Clients* includes enhancements in the JMX API.

Known Issues and Limitations

This section contains a list of the known issues with Message Queue 4.3. The following product areas are covered:

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 version 4.3.

Product Registry and Java ES

Message Queue 4.3, like Message Queue 4.2 and 4.1, is installed by a relatively new installer, which also installs and upgrades the Java Enterprise System (Java ES) shared components required by Message Queue; for example, JDK, NSS, JavaHelp, and so on.

The new Message Queue installer and the older Java ES installer, which was used to install previous Message Queue versions, do not share the same product registry. If a version of Message Queue that was installed with the Java ES installer is removed and then Message Queue 4.3 is subsequently installed by the Message Queue installer, the Java ES product registry might be in an inconsistent state. As a result, if the Java ES uninstaller is run, it may inadvertently remove Message Queue 4.3 and the shared components upon which it depends, even though it did not install them.

The best way to upgrade Message Queue software that was installed by the Java ES installer is as follows.

1. Use the Java ES uninstaller to remove Message Queue and its shared components.
2. Use the Message Queue installer to install Message Queue 4.3.

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.

- When a computer system has older versions of Message Queue and NSS/NSPR, the installer's Upgrade screen only lists Message Queue as requiring upgrade; it does not mention that NSS and NSPR need to be upgraded as well. All the relevant software will nevertheless be upgraded (as indicated by The ReadyToInstall screen which shows the correct information). *(Bug 6580696)*
- List of JDKs on JDK Selection Screen is active even when “Choose a JDK” option is not selected. *(Bug 6650874)*
- When using the Message Queue uninstaller, clicking Cancel instead of Remove will still remove some installer files, and future uninstalls will fail. *(Bug 6760416)*
- 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 4.3 installer on a computer in which Open Message Queue has been previously installed, you might see the following warning message: "Error reading previous session data from Config-State." *(Bug 6764305)*

  Workaround: The message is benign and will not recur once the install has been completed. Or you can remove the `/var/install/config/mq/InstallDirectory.xcu` file to avoid the message.
- When running the Message Queue zip-based 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)*
- Trying to register Message Queue in text mode (`installer -t -r`) without having installed Message Queue throws a NullPtr Exception. *(Bug 6760991)*

  Workaround: Install Message Queue before trying to register the installation.

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 *Sun Java System Message Queue 4.3 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)*

- Attempting to run the installer in text mode (`installer –t`) on Windows causes an error message that is displayed in English even when the installer is run in non-English. Text mode is not supported on Windows. *(Bug 6594142)*

- 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)*
- When running the installer in text mode, when entering a username or password for Sun Connect registration or creating an online account, you cannot correct a user name or password using the backspace key. *(Bug 6673460)*

  Workaround: Use the `Control-H` keys instead of the backspace key, or use a different terminal emulator like `dtterm` or `xterm`.
- The Upgrade screen on the installer does not always correctly report the existing installed version of Message Queue or of the installer engine. *(Bug 6679765)*
- When using the installer in text mode and attempting Sun Connection registration with an invalid user and password names, the installer displays an "unable to register" dialog, throws a Null pointer exception, and exits. *(Bug 6666365)*

Installing on Linux

The following issues affect installation on the Linux Platform:

- On Redhat 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`*`..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 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)*
- Running the Message Queue installer in text mode on 64-bit Linux does not work. *(Bug 6771303)*

  Workaround: If you are trying to install remotely from a terminal window then you will have to use some remote display software to run the installer in GUI mode instead.

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 1–11 Version String Translation

Version as Displayed by the Installer on Solaris OS	Corresponding Message Queue Release
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

TABLE 1–11 Version String Translation *(Continued)*

Version as Displayed by the Installer on Solaris OS	Corresponding Message Queue Release
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)*
- Text mode (`installer –t`) is not supported on Windows. Running the installer in text mode on Windows will cause an error message to be displayed. This message is not localized when the installer is run in non-English locales. *(Bug 6594142)*

- 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=true
  imq.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)*

Broker Issues

The following issues affect the Message Queue broker.

- 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 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 "High-Availability Clusters: JDBC Configuration Properties " in *Sun Java System Message Queue 4.3 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=
  CREATE 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=
  CREATE INDEX &(index) ON $(name) (MESSAAGE_ID)
  ```

Broker Clusters

The following issues affect broker clusters.

- 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 QueueDatabase 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 *Sun Java System Message Queue 4.3 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

Sun Java System Message Queue 4.3 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`
- `jaas.jar`
- `jms.jar`
- `libmqcrt.so` (HPUX)
- `libmqcrt.so` (UNIX)
- `mqcrt1.dll` (Windows)

In addition, you can also redistribute the `LICENSE` and `COPYRIGHT` files.

Accessibility Features for People With Disabilities

To obtain accessibility features that have been released since the publishing of this media, consult Section 508 product assessments (available from Sun upon request) to determine which versions are best suited for deploying accessible solutions. Updated versions of applications can be found at `http://sun.com/software/javaenterprisesystem/get.html`.

For information on Sun's commitment to accessibility, visit `http://sun.com/access`.

How to Report Problems and Provide Feedback

If you have problems with Sun Java System Message Queue, contact Sun customer support using one of the following mechanisms:

- Sun Software Support services online at `http://www.sun.com/service/sunone/software`.

  This site has links to the Knowledge Base, Online Support Center, and ProductTracker, as well as to maintenance programs and support contact numbers.
- The telephone dispatch number associated with your maintenance contract.

So that we can best assist you in resolving problems, please have the following information available when you contact support:

- Description of the problem, including the situation where the problem occurs and its impact on your operation.
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem.
- Detailed steps on the methods you have used to reproduce the problem.
- Any error logs or core dumps.

Sun Java System Software Forum

There is a Sun Java System Message Queue forum available at the following location:

`http://swforum.sun.com/jive/forum.jspa?forumID=24`

We welcome your participation.

Java Technology Forum

There is a JMS forum in the Java Technology Forums that might be of interest.

`http://forum.java.sun.com`

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is Sun Java System Message Queue 4.3 Release Notes, and the part number is 820-6361.

Additional Sun Resources

Useful Sun Java System information can be found at the following Internet locations:

- Documentation

  `http://docs.sun.com/prod/java.sys`
- Professional Services

  `http://www.sun.com/service/sunps/sunone`
- Software Products and Service

  `http://www.sun.com/software`
- Software Support Services

  `http://www.sun.com/service/sunone/software`
- Support and Knowledge Base

  `http://www.sun.com/service/support/software`

- Sun Support and Training Services

  http://training.sun.com
- Consulting and Professional Services

  http://www.sun.com/service/sunps/sunone
- Developer Information

  http://developers.sun.com
- Sun Developer Support Services

  http://www.sun.com/developers/support
- Software Training

  http://www.sun.com/software/training