Sun Java System Message Queue 3.7 Update 2 Release Notes

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

Part No: 820–2382–10
September 2007

070914@18741

Contents

◆ ◆ ◆ CHAPTER 1

Sun Java System Message Queue 3.7 UR2 Release Notes

Version 3.7 UR2

Part Number 820-2382

These release notes contain important information available at the time of release of Sun Java™ System Message Queue 3.7 UR2. New features and enhancements, known issues and limitations, and other information are addressed here. Read this document before you begin using Message Queue.

The most up-to-date version of these release notes can be found at the Sun Java System Message Queue documentation web site. 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.

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

TABLE 1–1 Revision History

Date	Description of Changes
August 2006	Initial release of this document, describing 3.7 UR1 release.
February 2007	Final release of this document, describing 3.7 UR1 release.
September 2007	Final release of this document, describing 3.7 UR2 release.

About Message Queue 3.7 UR2

Sun Java System Message Queue is a full-featured message service that provides reliable, asynchronous messaging conformant to 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 3.7 UR2 is a bug fix release to Message Queue 3.7 UR1. These Release Notes describe known and fixed bugs, and a small number of documentation notes. This section includes the following information:

Features and Documentation Updates

This section describes minor features introduced in 3.7 UR2 and some documentation updates.

- "3.7 UR 2 Documentation" on page 6
- "Installation Information" on page 7
- "Transaction Management" on page 7
- "Fixed Ports for C Client Connections" on page 7

3.7 UR 2 Documentation

These Release Notes are the only document that has been updated for the 3.7 UR2 release. Otherwise, the 3.7 UR1 documentation is up to date and valid for use with the 3.7 UR2 release, except that there are some minor updates to the Installation Guide, which are noted in the next section.

Installation Information

The information in the 3.7 UR1 Message Queue Installation Guide is up to date for the 3.7 UR2 product, with the following exceptions.

- The name of the download files for the 3.7 UR2 product contains the string`mq3_7ur2`.
- The 3.7 UR2 product supports Java Runtime Environment (JRE) 1.6.0_02 and Java Development Kit (JDK) 1.5.0–12.
- The disk space required (on all platforms) for installation is approximately 250 MB.

Transaction Management

The following sections address issues in managing transactions.

Storing Transaction Data and Performance

To improve performance, Message Queue brokers are configured by default to use a memory-mapped file to store transaction data. On file systems that do not support memory-mapped files, you can disable this behavior by setting the broker property `imq.persist.file.transaction.memorymappedfile.enabled` to `false`.

Rolling Back Transactions

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 the broker administrator. In Message Queue 3.7 UR1 or 3. 7 UR2 you can use the `imqcmd` utility to clean up (roll back) transactions that are in the following states: `STARTED`, `FAILED`, `INCOMPLETE`, `COMPLETE`, `PREPARED`.

To help you determine whether a particular transaction can be rolled back (especially when it is not in a `PREPARED` state), the `imqcmd` 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, the 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

C clients can use the `MQ_SERVICE_PORT_PROPERTY` connection property to specify a fixed port to connect to. This can be useful if you're trying to get through a firewall or if you need to bypass the broker's port mapper service (which assigns ports dynamically).

Remember that you need to configure the JMS service port on the broker side as well. 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 `MQ_SERVICE_PORT_PROPERTY` to 1756 and set the `MQ_CONNECTION_TYPE_PROPERTY` to `SSL`.
- On the broker side: Set the imq.`serviceNameType.protocol`.port property to 1756 as follows.

```
imq.ssljms.ssl.port=1756
```

Hardware and Software Requirements

On the Solaris platform, the supported compiler has changed to Sun Studio 11 or later.

For additional information about hardware and software requirements, see *Sun Java System Message Queue 3.7 UR1 Installation Guide*.

About Message Queue 3.7 UR1

Message Queue 3.7 UR1 is a maintenance release to Message Queue 3.6. It includes bug fixes and a small number of minor enhancements. This section includes the following information:

- "What's New in This Release" on page 8
- "Hardware and Software Requirements" on page 11
- "Working with Solaris 10 Zones" on page 11

What's New in This Release

Message Queue 3.7 UR1 includes the following new features:

- "Combining Platform and Enterprise Features in One Edition" on page 8
- "Interface Changes to the C-API and C Client Runtime" on page 10
- "Persistent Store Format Changes" on page 11

These are described in the following subsections.

Combining Platform and Enterprise Features in One Edition

In an effort to streamline our product delivery, we are combining the Platform and Enterprise Edition of Sun Java Message Queue. Beginning with Message Queue 3.7 UR1, there will only be a single edition available, which effectively removes the feature restrictions in the stand-alone distribution. We hope this will simplify your experience with this product.

Combining editions also aligns Message Queue better with Solaris Enterprise System and provides a perpetual broad-based right to use Enterprise Edition features, with no support, maintenance, or indemnity. As with previous releases, we will continue to offer several licensing options for support and maintenance services. Message Queue will continue to be packaged with Java Enterprise System and the Application Platform Suite. Please check the online store at `http://www.sun.com` or consult your sales representative to find an option that best suits your needs. The next table describes the upgrade paths to the new single edition of Message Queue.

TABLE 1–2 Upgrade Paths for Message Queue 3.7 UR1

Prior Edition	Upgrade Path	Comments
Platform Edition	Sun Java System Message Queue 3.7 UR1	All features (Platform and Enterprise) are now available to 3.7 UR1 customers. Support options are available with license purchase.
Enterprise Edition	Sun Java System Message Queue 3.7 UR1	No feature changes. A range of licensing and support options are available.
Platform Edition support contracts	Upgrade to Enterprise Edition support contract	Existing support contracts for prior releases of Platform Edition will continue to be renewed. No new Platform Edition contracts will be issued for prior Platform Edition releases.
Enterprise Edition support contracts	No change	Existing contracts will continue to be renewed. New contracts will be issued.

The following table describes the changes in delivery sources for various Message Queue products.

TABLE 1–3 Changes in Delivery Sources for Message Queue Products

Product	Prior Delivery Source	New Delivery Source	Comments
Open Message Queue	Not applicable	Sun download center product page	Standalone download. Community support only. No support contracts available.
Message Queue Platform Edition	Sun download center via the Message Queue product page	No longer available	Only the single edition of Message Queue that combines Platform and Enterprise features is now available.

TABLE 1–3 Changes in Delivery Sources for Message Queue Products *(Continued)*

Message Queue Enterprise Edition trial (via Platform Edition)	Sun download center via the Message Queue product page	Trial license no longer needed	No longer needed
Message Queue Enterprise Edition 90 day Trial (via Java Enterprise System download or CD)	Java Enterprise System download center, prior to version 3 GA (March 2006)	Solaris Enterprise System download center	Solaris Enterprise System license. Support options are available with product license. (A 90 day trial license is no longer needed.)
Message Queue Enterprise Edition via SunStore, CD, individual license, Java Enterprise System license, Suite license, delivered via Java Enterprise System	Java Enterprise System or Suite download center, media	Solaris Enterprise System or Suite download center, media fulfillment	No change.

Interface Changes to the C-API and C Client Runtime

- New function: `MQGetDestinationName()`

  ```
  MQGetDestinationName (const MQDestinationHandle destinationHandle,
                             MQString * destinationName);
  ```

  Use this function to get the name of a destination. The returned `destinationName` is a copy that the caller is responsible for freeing by calling the `MQFreeString()` function.

  Parameters

destinationHandle	A handle to the destination whose name you want to know.
destinationName	The output parameter for the name.

  This function is useful when using the reply-to pattern. You can use the `MQGetMessageReplyTo` function to obtain a handle to the destination where the message should be sent. You can then use the `MQGetDestinationName` to get the name of that destination. Having obtained the destination name, you can do message processing based on the name.

- New enumerated value: `MQ_MESSAGE`

  The new `MQMessageType`, `MQ_MESSAGE`, allows C clients to exchange JMS messages of type `Message` with other Message Queue clients (both C and Java):

  ```
  typedef enum _MQMessageType {MQ_TEXT_MESSAGE = 0,
                               MQ_BYTES_MESSAGE = 1,
                               MQ_MESSAGE = 3,
                               MQ_UNSUPPORTED_MESSAGE = 2} MQMessageType;
  ```

The `MQ_MESSAGE` type identifies messages that have a header and properties but no message body. You use the `MQCreateMessage()` function to create a message of this type.

- A new connection property, `MQ_UPDATE_RELEASE_PROPERTY`, that specifies the update release version for the installed version of Message Queue. Use the `MQGetMetaData()` function to obtain version information.

Persistent Store Format Changes

Two changes to the Message Queue persistent store format have been made to improve performance. One change is to the file store, the other to the JDBC store.

- Transaction Information in the file store

  The format of transaction state information stored in the Message Queue file-based persistent store has been changed to reduce disk I/O and improve the performance of JMS transactions.
- Oracle JDBC Store

  In previous versions of Message Queue, the store schema used with Oracle used the `LONG RAW` data type to store message data. In Oracle 8, Oracle introduced the `BLOB` data types and deprecated the `LONG RAW` type. Message Queue 3.7 UR1 switches to the `BLOB` data type to improve performance and supportability.

Because these changes impact store compatibility, the store version has been changed from 350 to 370. Message Queue 3.7 UR1 supports automatic conversion of the persistent store from the older 200 and 350 versions to the 370 version - both for JDBC and for file based stores. The first time `imqbrokerd` starts, if the utility detects an older store it will migrate the store to the new format, leaving the old store behind.

If you should need to roll back this upgrade, you can uninstall Message Queue 3.7 UR1 and then reinstall the version you were previously running. Since the older copy of the store is left intact, the broker can run with the older copy of the store.

Hardware and Software Requirements

Hardware and software requirements for Message Queue are provided in the *Sun Java Enterprise System Installation Guide*.

Working with Solaris 10 Zones

A *zone* is a Solaris Container technology that provides separate environments on a machine and logically isolates applications from one another. Zones allow you to create virtual operating system environments within an instance of the Solaris operating system. Running applications in different zones allows you to run different instances or different versions of the same application on the same machine while, at the same time, permitting centralized administration and efficient sharing of resources.

This section provides a brief description of zones and describes their use with Message Queue 3.7 UR1.

Zones Basics

A zone environment includes a global zone and one or more non-global zones. When Solaris 10 is first installed on a system there is only one global zone. An administrator can create other non-global zones as children of the global zone. Each zone appears as an independent system running Solaris. Each zone has its own IP address, own system configuration, own instances of running applications, and its own area on the file system.

The global zone contains resources that can be shared among non global zones; this allows the centralization of certain administrative functions. For example, packages installed in the global zone are available (propagated) to all existing non-global zones. This enables you to centralize life-cycle management like installation, upgrade, and uninstallation. At the same time, the isolation provided by non-global zones results in greater security and allows you to have differently configured instances or different versions of the same application running on the same machine.

Non-global zones are either whole root zones or sparse root zones: which of these you choose as an environment for an application depends on how you want to balance administrative control with resource optimization.

- *Whole root zones* contain a read/write copy of the file system on the global zone. Packages installed in the global zone are automatically copied (with their registry information) to the whole root zones. This maximizes administrative control, at the expense of resources.
- *Sparse root zones* contain a read/write copy of a portion of the file system on the global zone; other file systems are mounted as read-only file systems. Packages installed in the global zone are available to sparse root zones by means of read-only file systems and through the automatic synchronization of registry information. Sparse root zones optimize resource sharing at the cost of centralized administration.

Java Enterprise System Zones Limitations

The components that make up the Java Enterprise System depend on some shared components; this creates some limitations in working with zones. In a zones environment, shared components are governed by the following rules.

- All shared components within a zone must be of the same JES version. This requirement has three consequences.

  - If you want to install different versions of shared components, each version must reside in a separate zone.
  - Within a zone, if a shared component is upgraded or a later version is installed then all shared components must be upgraded.
  - When you install shared components in the global zone, you must take care that shared components in non global zones are upgraded if necessary.
- Shared components cannot be installed in sparse root zones because of the read/only file system in sparse root zones. Instead, they must be installed in the global zone. Those product components that depend on shared components must first be installed in the global zone and then propagated into non-global zones.

These requirements affect the installation of Message Queue because it is a component product of Java Enterprise System and, as such, is limited in its use of zones.

Note – The Message Queue product is installed into the `/usr` directory and must therefore be installed or upgraded in the global zone first.

Message Queue Cases

When Message Queue is installed in the global zone, it is set to propagate into all of the non-global zones. After installing Message Queue in the global zone, you will have the same version of Message Queue installed in all zones: if you log into any zone and run the command `pkginfo -l SUNWiqu`, you will see it installed, and it will be the same version as in the global zone. You can then run independent instances of the Message Queue broker in each zone since they do not share the instance and configuration data kept in the `/var` and `/etc` directories. (Most other Java Enterprise System components are not propagated if they are installed in the global zone.)

Because Message Queue is propagated into non-global zones, the global instance is forever linked to the installations in the non-global zones. Therefore, any time you uninstall or upgrade Message Queue in the global zone it will impact instances running in the non-global zones. The following example shows how this might cause unintended results.

1. You install Message Queue 3.7 UR1 in the global zone. This results in the Message Queue 3.7 UR1 packages also being installed into all non-global zones.
2. You uninstall Message Queue 3.7 UR1 in a whole root zone. Then, you install Message Queue 3.6 in the whole root zone.

   You now have different versions of Message Queue running in different zones, which is a set up you might find useful.
3. You uninstall Message Queue 3.7 UR1 in the global zone. This will uninstall Message Queue from all other zones - including the Message Queue 3.6 instance in the whole root zone.

Always be aware of the cascading effect of installing or uninstalling Message Queue in the global zone.

The following two use-cases explain how you install different instances and different versions of Message Queue in different zones.

Note – If you want to install Message Queue in a whole root zone on Solaris 10, Solaris 10U1, or Solaris 10U2, you must upgrade Lockhart in the global zone first. See the workaround for bug 645030 for additional information.

▼ To Install the Same Version of Message Queue in Different Zones

1. **Install the desired version of Message Queue in the global zone.**

   These versions will be propagated into any existing non-global zone. If you create additional non-global zones, Message Queue will also be propagated into these zones. (You can install different instances in whole root zones as well as sparse root zones, but using sparse root zones allows you to make more efficient use of disk space and other resources).

2. **If you want Message Queue to be propagated into any other non-global zones, create these zones now.**

3. **Run an instance of Message Queue in each non-global zone.**

▼ To Install Different Versions of Message Queue in Different Zones

1. **Uninstall Message Queue from the global zone.**

2. **Create whole root zones and configure each zone not to share the /usr directory by using the following directive when you create the zone**

   ```
   remove inherit-pkg-dir dir=/usr
   ```

3. **Install different versions of Message Queue in each whole root zone.**

Note – Remember that installing or uninstalling Message Queue in the global zone will affect all instances (and versions) of Message Queue running in whole root zones.

Bugs Fixed in This Release

This section describes the bugs fixed in Message Queue 3.7 UR2 and 3.7 UR1.

TABLE 1–4 Fixed Bugs in Message Queue 3.7 UR2

Bug Number	Description
6401169	The `imqcmd commit` and `rollback` commands do not prompt for confirmation.
6448939	Message loss when broker that an Message Driven Bean is consuming from is killed midway through a transaction.
6467874	Broker exception on restart if the broker was killed after a commit and before acknowledged message was removed.

TABLE 1–5 Fixed Bugs in Message Queue 3.7 UR1

Bug Number	Description
6193884	Message Queue outputs garbage message to syslog in locales that require non-ASCII characters to display messages.
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.
6284769	The QueueBrowser leaks memory even when a new browser is created and closed for each enumeration.
6294767	Message Queue broker needs to set `SO_REUSEADDR` on the network sockets it opens.
6304043	The broker does not validate that a `clientID` is not null for shared nondurable subscriptions.
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 1–5 Fixed Bugs in Message Queue 3.7 UR1 *(Continued)*

Bug Number	Description
6321117	Multi-broker cluster initialization throws `java.lang.NullPointerException`.
6330053	The jms client runtime throws `java.lang.NoClassDefFoundError` when committing a transaction from the subscriber.
6340250	Support `MESSAGE` type in C-API.
6351293	Add Support for Derby database.
6381693	The `JMSRedelivered` flag is set to false for messages redelivered to a remote consumer after the consumer's broker restarts.
6388049	Cannot clean up an incomplete XA transaction.
6403968	Add consumer-based flow control protocol to allow load balancing for multiple receivers
6403958	A broker exception is raised when the dead message queue tries to remove the oldest non-persistent message.
6406862	Not possible to monitor any destinations after broker error `Monitor destination errormq.metrics.destination.queue.Name`
6415068	Transaction recovery fails in AS 8.1 UR2, generic resource adapter for JMS Message Queue.
6421781	Support connection to broker using the following syntax `–b mqtcp://hostName:portNumber/serviceName`
6423696	`Session.rollback` does not actually roll back consumed messages after COMMIT REPLY error.

Important Information

This section contains the latest information that is not contained in the core product documentation. This section covers the following topics:

- "Installation Notes" on page 16
- "Compatibility Issues" on page 17
- "Documentation Updates for Message Queue 3.7 UR1" on page 17

Installation Notes

Refer to the *Sun Java System Message Queue 3.7 UR1 Installation Guide* for information about pre-installation instructions, upgrade procedures, and all other information relevant to installing Message Queue as a standalone product on the Solaris, Linux, and Windows platforms.

Refer to the *Sun Java Enterprise System Installation Guide* for information about pre-installation instructions and all other information relevant to installing the Message Queue product (bundled with Java Enterprise System) on the Solaris, Linux, and HPUX platforms.

Refer to the *Sun Java Enterprise System Upgrade and Migration Guide* for information about upgrade and migration instructions relevant to upgrading to Message Queue (as part of Java Enterprise System) on the Solaris, Linux, HPUX, and Windows platforms.

Compatibility Issues

This section covers compatibility issues in Message Queue 3.7 UR1.

Interface Stability

Message Queue uses many interfaces that may change over time. Appendix B, "Stability of Message Queue Interfaces," in *Sun Java System Message Queue 3.7 UR1 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 may introduce changes that make your clients incompatible with that release. This information is provided now to allow you to prepare for these changes.

- 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.
- 3.5 and earlier brokers may no longer be able to operate in a cluster with newer brokers.
- In future releases Message Queue clients may not be able to use JDK versions that are earlier than 1.3.
- Support for SOAP administered objects will be discontinued in future releases of Message Queue.

Documentation Updates for Message Queue 3.7 UR1

The following sections describe updates and changes to Message Queue 3.7 UR1 documentation, other than minor corrections and additions. These documents are also valid for the 3.7 UR2 release, except for the 3.7 UR1 Release Notes, which are superseded by the 3.7 UR2 Release Notes.

Developer’s Guide for C Clients

The *Sun Java System Message Queue 3.7 UR1 Developer’s Guide for C Clients* was updated to reflect the addition of the `MQGetDestinationName` function, of the `MQ_Message` message type, and of the connection property `MQ_UPDATE_RELEASE_PROPERTY`.

Developer's Guide for Java Clients

The *Sun Java System Message Queue 3.7 UR1 Developer’s Guide for Java Clients* was updated to include information about setting up secure clients and about how clients should handle exceptions that occur during automatic reconnection.

Administration Guide

The *Sun Java System Message Queue 3.7 UR1 Administration Guide* was updated to provide information about the broker's database tables and about configuring the broker to use fixed ports.

Known Issues and Limitations

This section contains a list of the known issues with Message Queue 3.7 UR1 and 3.7 UR2. The following product areas are covered:

- “General Issues” on page 19
- “LDAP User Repository Properties” on page 19
- “Broker Clusters” on page 19
- “Administration/Configuration” on page 20
- “Broker Issues” on page 21

For a list of current bugs, their status, and workaround, 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`.

General Issues

- If you are installing standalone Message Queue 3.7 UR2 (from a zipped file) please note that the broker cannot be automatically installed as a Windows service. To install the broker as a Windows service, use the `imqsvcadmin install` command.
- A connection service using SSL is currently limited to supporting only self-signed server certificates, that is, host-trusted mode.
- When a JMS client using the HTTP transport 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.

LDAP User Repository Properties

In Message Queue 3.7 UR1, the example broker configuration for using an LDAP server as a user repository is provided in the comment area in the `config.properties` file, and the LDAP user repository example in the `default.properties` file has been commented out.

If you previously relied on any property value in the example LDAP user repository properties specified in the `default.properties` file, your JMS application client will receive a security exception when attempting to create a JMS connection. This will happen after you upgrade to Message Queue 3.7 UR1.

When your JMS client tries to make a connection to the Message Queue 3.7 UR1 broker, you will get a error in the broker log and your JMS client will receive the following exception:

```
SecurityException.
20/Aug/2004:11:16:41 PDT] ERROR [B4064]: Ldap repository ldap property
.uidattr not defined for authentication type
basic:com.sun.messaging.jmq.auth.LoginException:
[B4064]: Ldap repository ldap property .uidattr not defined
for authentication type basic
```

Workaround Set the broker property `imq.user_repository.ldap.uidattr` following the instructions in Chapter 7, "Managing Security," in *Sun Java System Message Queue 3.7 UR1 Administration Guide*.

Broker Clusters

The following items relate to the use of 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 using the `imqbrokerd -cluster` command line argument, be careful to ensure that all brokers in the cluster are included.
- A client connected to a broker that is part of a cluster cannot currently use `QueueBrowser` to browse queues that are located on remote brokers in that cluster. The client can only browse the contents of queues that are located on the broker to which it is directly connected. The client may still send messages to any queue or consume messages from any queue on any broker in the cluster; the limitation only affects browsing.
- If a master broker is not used in a broker cluster, persistent information stored by a broker being added to the cluster is not propagated to other brokers in the cluster.
- Connection is dropped for a broker in a cluster (*Bug ID 6377527*).

  One reason this can happen is that the broker address (whose connection is dropped) is resolved to be the loopback IP address (127.0.0.1).

  Workaround Make sure that the broker address does not resolve to the loopback IP address.
- In a broker cluster, a broker will queue messages to a remote connection which has not been started (*Bug ID 4951010*).

  *Workaround*The messages will be received by the consumer once the connection is started. The messages will be redelivered to another consumer if the consumer's connection is closed.

Administration/Configuration

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

- The `imqadmin` and `imqobjmgr` utilities throw an error when the `CLASSPATH` contains double quotes on Windows machines (*Bug ID 5060769*)

  Workaround You can ignore this error message; the broker correctly handles notifying consumers of any error. This error does not affect the reliability of the system.
- The `-javahome` option in all Solaris and Windows scripts does not work if the value provided contains a space (*Bug ID 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 destination. Note that the client application will always get all the messages on the queue. Thus, the `imqQueueBrowserMaxMessagesPerRetrieve` attribute affects how the queued messages are chunked, to be delivered to the client runtime (fewer large chunks, or more smaller chunks), but it does not affect the total messages browsed. Changing the value of this attribute might impact performance, but it will not result in the client application getting more or less data (*Bug ID 6387631*).

Broker Issues

The following issues affect the Message Queue broker.

- The `imqbrokerd –license` command displays outdated or duplicate information. It displays information about the try license, even though this type of license is no longer supported (*Bug ID 6489711*) and it displays duplicate information about the unl license (*Bug ID 6441015*).

  Workaround These are cosmetic problems that require no workaround.
- The broker does not honor the default limit of 1000 messages for the dead message queue; it keeps adding messages to the dead message queue until the broker runs out of memory. (*Bug ID 6502744*)

  Workaround Reset the Dead Message Queue limit to 1001 or any value other than 1000.
- HTTPS `createQueueConnection` occasionally throws exception on Windows 2000. (*Bug ID 4953348*).

  Workaround Retry the connection.
- When using Ctrl-C to shut down broker, transactions may be cleaned up after store is closed (*Bug ID 4934446*).

  The broker may show errors with the following reason “Store method accessed after the store is closed.” if the broker is shut down while messages or transactions are processed.

  Workaround You can ignore this error message; the broker correctly handles notifying consumers of any error. This error does not affect the reliability of the system.
- Broker becomes inaccessible when persistent store opens too many destinations. (*Bug ID 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 ID 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).

  Workaround There is no workaround for this problem.

- Message selection using `JMSMessageID` doesn't work (*Bug ID 6196233*).

  Workaround Change the selector from the following expression

  `JMSMessageID = "ID:message-id-string"`

  To the following expression

  `JMSMessageID IN ('ID:message-id-string', 'message-id-string')`

- Message Queue A queue browser shows uncommitted messages (*Bug ID 6264003*).

  When browsing the contents of a queue, messages that were produced in a transaction but are not yet committed may appear in the queue browser enumeration.

  Workaround There is no workaround for this problem.

- Messages may become unavailable after broker crashes during a commit (*Bug ID 6467874*).

  In very rare cases, during a broker crash, messages in a transaction may become unavailable to consumers. Specifically, there is a small window during Commit processing that may cause the message to become stuck in the persistent store. When this occurs, the following message is displayed at startup of the broker after a crash.

  ```
  [06/Sep/2006:10:11:11 PDT] ERROR [B2085]:
     Loading Destination q0 [Queue] failed.
     Messages stored on that destination will not be available.:
     > com.sun.messaging.jmq.jmsserver.util.BrokerException:
    The message 8-129.145.180.87(b8:8b:26:15:41:26)-38998-1157562551217
    has an associated acknowledgement list already.
  ```

  Workaround There is no workaround for this problem.

Redistributable Files

Sun Java System Message Queue 3.7 UR2 contains the following set of files that you may use and distribute in binary form:

`jms.jar`	`libmqcrt.so` (UNIX)
`imq.jar`	`libmqcrt.so` (HPUX)
`imqxm.jar`	`mqcrt1.dll` (Windows)
`fscontext.jar`	

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 3.7 UR2 Release Notes, and the part number is 820-2382.

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