Sun™ ONE Message Queue Release Notes

Version 3.0.1 SP2

Part Number 817-3731-10

August 2003

These release notes contain important information available at the time of release of Version 3.0.1 Service Pack 2 (SP2), of Sun™ Open Net Environment (Sun ONE) Message Queue (MQ). New features and enhancements, known limitations and problems, technical notes, and other information are addressed here. Read this document before you begin using MQ 3.0.1 SP2.

The most up-to-date version of these release notes can be found at the Sun ONE documentation web site: http://docs.sun.com/?p=/coll/S1_MessageQueue_301. 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 manuals.

These release notes contain the following sections:

Revision History

Table 1 Revision History

Date	Description of Changes
August 2003	The following changes were made to this document: • Slight changes were made in the organization of the document and in clarification of version numbers. • Added new section, What's New in Message Queue 3.0.1 SP2. • Added bugs 4879448, 4883126, 4888270, and 4883126 to Known Bugs. • Added bugs 4683129, 4735757, 4758424, 4758427, 4770212, 4770518, 4809079, 4821708, 4828860, 4835586, 4879448 to Bugs Fixed in Message Queue 3.0.1 Versions.
October 2002	Initial release of this document

Java™ Message Service (JMS) Compliance

MQ 3.0.1 versions have been certified as compliant with the Java Message Service (JMS) 1.1 specification: it has passed the JMS 1.1 Compatibility Test Suite (CTS).

MQ 3.0.1, which is the native JMS provider for Sun ONE Application Server 7, has also passed the J2EE 1.3.1 CTS testing of the Sun ONE Application Server 7 (which requires compliance with JMS 1.0.2b)

What's New in Message Queue 3.0.1 SP2

MQ 3.0.1 SP2 is an update of MQ 3.0.1 SP1 (which was a bug release with no new features). MQ 3.0.1 SP1 was an update of MQ 3.0.1. In these Release Notes, references to 3.0.1 versions generally mean 3.0.1, 3.0.1 SP1, and 3.0.1 SP2 respectively.

The MQ 3.0.1 SP2 product includes all the new features of MQ 3.0 and 3.0.1, plus two additional 3.0.1 SP2 features.

Features New in Message Queue 3.0.1 SP2

MQ 3.0.1 SP2 includes two new capabilities as compared to MQ 3.0.1:

- Changes to the Linux installation of MQ from a tar-based install to a Red Hat Package Manager (RPM) install.

  When using the RPM install command, a record of everything that you install is placed in an installation registry so that you can track all changes to the product.

- Substantial performance improvement for consumers using selectors. Performance improvement increases as the number of consumers increases.

Features New in Message Queue 3.0.1

MQ 3.0.1 includes two new capabilities as compared to MQ 3.0:

- Substantial performance boost

  MQ 3.0.1 provides message delivery throughput up to double that attained with MQ 3.0, a performance boost that is especially important under heavy load conditions. The improvement was achieved through changes in the marshalling of JMS message properties for transmission over the wire.

- Certified for Sun ONE Application Server 7.0

  MQ 3.0.1 is certified for Sun ONE Application Server 7.0, and is used as its native JMS provider. MQ has been integrated with the Application Server, providing JMS messaging support in an Application Server environment. You can configure the system for an internal MQ message server managed with Application Server administration tools, or an external MQ message server requiring MQ administration tools.

- Support for Linux Red Hat 7.2 (JDK 1.4.1)

  MQ 3.0.1 is now certified for JDK 1.4.1 on Linux Red Hat 7.2 (and still supported on Linux Red Hat 7.1).

Features New in Message Queue 3.0

MQ 3.0 included a number of changes to Version 2.0 of the product—iMQ 2.0 (and iMQ 2.0, Service Pack 1).

Notable among these changes is that the product is now available in two editions: a Platform Edition and an Enterprise Edition.

Platform Edition Provides basic JMS support, and is best suited to small-scare deployments and development environments

Enterprise Edition Provides HTTP/HTTPS support, enhanced scalability, and security features, and is best suited to large-scale deployments.

(See the introduction to the MQ *Administrator's Guide* or the MQ *Developer's Guide* for more information on these editions.)

The descriptions, below, of changes in the MQ 3.0.1 product are grouped according to whether they apply to both editions or to the Enterprise Edition only.

Both Enterprise and Platform Editions

- Support for distributed transactions

  MQ now supports the JTA XA Resource API, meaning that production and consumption of messages can be part of a larger distributed transaction involving other resource managers, such as database managers (see Chapter 1 of the MQ *Developer's Guide*). This feature is also supported with administrative tools for managing transactions (see Table 6-12 of the MQ *Administrator's Guide*). Programming information and examples are not yet available in the MQ 3.0.1 release product.

- Support for JMS 1.1

  MQ now supports the added features of the JMS 1.1 specification, which provides a simplified approach to JMS client programming as compared to JMS 1.0.2. In particular, a JMS client can perform both point-to-point and publish/subscribe messaging over the same connection and within the same session, and can include both queues and topics in the same transaction.

  In short, a JMS client developer need not make a choice between the separate point-to-point and publish/subscribe programming domains of JMS 1.0.2, opting instead for the simpler, unified domain approach of JMS 1.1. This is the preferred approach, however the JMS 1.1 specification continues to support the separate JMS 1.0.2 programming domains. (In fact, the example applications included with the MQ product as well as the code examples provided in the MQ *Developer's Guide* all use the separate JMS 1.0.2 programming domains.)

NOTE Developers of applications that run in the Sun ONE Application Server environment are limited to using the JMS 1.0.2 API. This is because the Application Server complies with the J2EE 1.3 specification, which supports only JMS 1.0.2. This means that any JMS messaging performed in servlets and EJBs (including message-driven beans) must be based on the domain-specific JMS APIs.

- Support for SOAP messaging using JAXM

  Supports creation and delivery of messages that conform to the Simple Object Access Protocol (SOAP) specification, using JAXM—the Java API for XML Messaging. SOAP allows for the exchange of structured XML data between peers in a distributed environment. MQ also supports the delivery of SOAP messages *via* JMS messaging. See the MQ *Developer's Guide* for more information.

- Improvements in persistent store

  MQ now provides more flexibility in balancing disk space and performance when using the built-in persistent store (see Table 2-5 of the MQ *Administrator's Guide*). Also, administrators now have the option of removing only messages or durable subscriptions from the persistent store when restarting a broker (see `reset` option in Table 5-2 of the MQ *Administrator's Guide*).

- Overriding JMS message header fields

  MQ now allows an administrator to better control message server resources by overriding the JMS message header fields that specify message persistence, priority, and expiration (see Table 4-4 of the MQ *Developer's Guide*).

- Improved management of durable subscriptions

  MQ now supports the purging of all messages for a specified durable subscription (see Table 6-11 of the MQ *Administrator's Guide*).

- New hostname configuration property

  MQ now supports more than one network interface card on a computer by letting administrators choose which hostname will be used by MQ connection services (see Table 2-3 of the MQ *Administrator's Guide*).

- Updating the default queue delivery policy

  MQ now allows an administrator to update the default delivery policy set for a queue destination (see Table 6-10 of the MQ *Administrator's Guide*).

- Support for Java 2 Platform, Standard Edition (J2SE) 1.4

  The broker and MQ administration tools are now supported on the Java Runtime Environment (JRE) 1.4, and JMS clients are now supported on the Java Software Development Kit (JDK) 1.4.

- New file system layout on the Solaris™ environment

  The installed directory structure of MQ 3.0.1 on Solaris has been changed to conform to general file system standards for the platform. MQ 3.0.1 files are no longer installed under a single root installation directory, but are dispersed to standard locations in the Solaris file system.

Enterprise Edition Only

- Support for secure HTTP (HTTPS)

  MQ now supports secure messaging over HTTP (see Appendix B of the MQ *Administrator's Guide*). This new connection service provides for encryption of messages from message producer through to message consumer (that is, from JMS client, through HTTPS tunnel servlet, to broker, and *visa versa*).

Message Queue Documentation Updates

The following MQ 3.0.1 and MQ 3.0.1 SP2 documents have been updated from Version 3.0 of the product. These updated documents can be found at the Sun ONE documentation web site: http://docs.sun.com/coll/S1_MessageQueue_301.

Installation Guide

The MQ *Installation Guide* has been updated to include changes to MQ 3.0.1 SP2 (see "Features New in Message Queue 3.0.1 SP2" on page 3).

Administrator's Guide

The MQ *Administrator's Guide* has been updated to include new features that are in MQ 3.0.1 (see "Features New in Message Queue 3.0.1" on page 3).

Developer's Guide

The MQ *Developer's Guide* has been updated to include new features that are in MQ 3.0.1 (see "Features New in Message Queue 3.0.1" on page 3).

Compatibility Issues

MQ 3.0.1 versions are fully compatible with MQ 3.0, and upgrading from MQ 3.0 to MQ 3.0.1 requires no changes in broker configuration, administered objects, administration tools, or client applications.

However, MQ 3.0.1 versions (hereafter in this section simply referred to as 3.0.1) are generally *not* compatible with iMQ 2.0. In particular, there are a number of issues that you might need to address when upgrading from iMQ 2.0 (or iMQ 2.0 Service Pack 1) to MQ 3.0.1:

- Broker Compatibility
- Administered Object Compatibility
- Administration Tool Compatibility
- Client Compatibility

Broker Compatibility

An MQ 3.0.1 broker will not inter-operate with an iMQ 2.0 broker due to changes in broker properties and in the persistent store schema. However, some iMQ 2.0 data is compatible with MQ 3.0.1, as shown in Table 2, and can be preserved when upgrading to MQ 3.0.1. When upgrading from iMQ 2.0 to MQ 3.0.1, you should consider the following:

- You can copy iMQ 2.0 `config.properties` files to another location and, in most cases, consult the property settings they contain when you configure MQ 3.0.1 brokers.
- Any persistent iMQ 2.0 data—messages, destinations, durable subscriptions—cannot be re-used. In particular, you will need to re-create iMQ 2.0 destinations in your MQ 3.0.1 brokers.
- You can continue to use iMQ 2.0 user repository and access control properties files after installing MQ 3.0.1. The MQ 3.0.1 installer does not overwrite these files. However, you will have to move them to the appropriate MQ 3.0.1 location (see Appendix D of MQ *Administrator's Guide*).

Table 2 Compatibility of MQ 3.0.1 with iMQ 2.0 Data

iMQ 2.0 Data Category	Location of iMQ 2.0 Data	Compatibility with MQ 3.0.1
Broker properties	`IMQ_VARHOME/stores/`*brokerName*`/props/config.properties`	Incompatible; do not use.
Persistent store (messages, destinations, durable subscriptions, transactions)	`IMQ_VARHOME/stores/`*brokerName*`/filestore/` or JDBC accessible data store	Incompatible; do not use.
Administered objects	local directory or LDAP server	Compatible; can use and/or convert to 3.0.1
Security: user repositories	`IMQ_VARHOME/security/passwd` or LDAP server	Compatible. Move to following location: `IMQ_HOME/etc/passwd` (`/etc/imq/passwd` on Solaris)
Security: access control file	`IMQ_VARHOME/security/accesscontrol.properties`	Compatible. Move to following location: `IMQ_HOME/etc/...` (`/etc/imq/...` on Solaris)

Administered Object Compatibility

MQ 3.0.1 administered objects have been enhanced with new attributes and iMQ 2.0 attributes have been renamed. Therefore, when upgrading from iMQ 2.0 to MQ 3.0.1, you should consider the following:

- You can use the same object store and administered objects that you created in iMQ 2.0; however, it is best to upgrade your administered objects after installing MQ 3.0.1. The Administration Console (`imqadmin`) and the ObjectManager command line utility (`imqobjmgr`), when performing an update operation, will convert iMQ 2.0 administered objects into MQ 3.0.1 administered objects.
- The MQ 3.0.1 client runtime will look up and instantiate iMQ 2.0 administered objects by converting them into local MQ 3.0.1 administered objects, but this will *not* convert iMQ 2.0 administered objects in the object store into MQ 3.0.1 administered objects.
- JMS clients (applications and/or components) that directly instantiate administered objects—that is, that are JMS provider-dependent—need to be rewritten to accommodate new administered object attribute names (see Chapter 4 and Appendix A of the MQ *Developer's Guide* for information on administered object attributes).

- Scripts that start JMS clients and which set administered object attribute values using command line options need to be rewritten to accommodate the new administered object attribute names (see Chapter 4 and Appendix A of the MQ *Developer's Guide* for information on administered object attributes).

Administration Tool Compatibility

Because of the renaming of many files and directories (specifically to replace the string "jmq" with "imq"), all MQ 3.0.1 command line utilities, broker properties, administered object attributes, and internal file names have changed. Therefore, when upgrading from iMQ 2.0 to MQ 3.0.1, you should consider the following:

- Any scripts that use command line utilities (`imqbrokerd`, `imqcmd`, `imqobjmgr`, and so forth) need to be edited to replace the old commands with the newly-named commands. Note, especially, that the `jmqbroker` command is now `imqbrokerd`.
- The Administration Console (`imqadmin`) allows you to manage several brokers and/or object stores concurrently, and saves the list of managed entities that are displayed in the navigational pane on the left side of the screen. Thus each time you launch the Console, the list of managed entities is re-displayed. The name of the directory in which user settings for the iMQ 2.0 Administration Console were stored has changed for MQ 3.0.1. If you wish to preserve the old Console settings when upgrading from iMQ 2.0 to MQ 3.0.1, you need to change the name of the directory where the `brokerlist.properties` and `objstorelist.properties` files are stored from *$HOME*`/.jmq/admin` to *$HOME*`/.imq/admin`, where *$HOME* is the Console user's home directory.

Client Compatibility

When upgrading from iMQ 2.0 to MQ 3.0.1, you should consider the following:

- An MQ 3.0.1 broker will support the iMQ 2.0 client runtime (but without additional MQ 3.0.1 capabilities), but an iMQ 2.0 broker will *not* support the MQ 3.0.1 client runtime.
- JMS clients built on JDK 1.2, 1.3, or 1.4 can inter-operate with a broker running JRE 1.4. However, clients that use a secure (SSL-based) connection to a broker will require additional JSSE and JNDI libraries if they are not built on JDK 1.4 (which includes these libraries).
- The JMS 1.1 API (supported by MQ 3.0.1) clarifies the behavior of the `Message.acknowledge()` method, used to acknowledge message consumption in `CLIENT_ACKNOWLEDGE` sessions. This might require you to modify existing JMS clients.

This `Message.acknowledge()` method now acknowledges *all* messages consumed in the session at the time the method is called. This change in behavior from the 1.0.2 API (supported by iMQ 2.0) is illustrated in the following example: suppose a client consumes four messages from a queue in the same session, say A, B, C, and D in that order, and all were consumed before the client calls the acknowledge method on message C.

- In 1.0.2, only messages A, B, and C, would get acknowledged since D was consumed after message C.
- In 1.1, all the messages (including D) are acknowledged since they were all consumed.

The acknowledgement is independent of the order in which messages are consumed, so long as they are consumed in the same session; or stated another way, the message on which the `acknowledge()` method is called no longer determines which messages get acknowledged.

- The JMS 1.1 API (supported by MQ) clarifies the use of the client identification used to keep track of durable subscriptions. This might require you to modify existing JMS clients.

  In iMQ 2.0, the behavior was to automatically set the ClientID to the local IP address of the client if a durable subscription was created without explicitly setting a ClientID value. In MQ 3.0.1, the behavior is to throw an exception if a durable subscription is created without explicitly setting a ClientID value. In other words a ClientID value must always be set—either in client code or using an attribute of the connection factory object—when durable subscriptions and durable connection consumers are used.

- When using message selectors in iMQ 2.0, a workaround was necessary to accommodate a bug that has been fixed in MQ 3.0.1. This might require you to modify existing JMS clients.

  In iMQ 2.0, if a string literal contained multi-byte characters, you had to use a double escape on Unicode characters (for example, `selector = “property = ‘\\u033e\\u033f’”`). In MQ 3.0.1, the normal representation for Unicode characters can be used (for example, `selector = “property = ‘\u033e\u033f’”`).

Known Limitations

Limitations shown in this section are grouped according to whether they apply to both Enterprise and Platform Editions of MQ 3.0.1 versions or to the Enterprise Edition only.

Both Enterprise and Platform Editions

- Windows platforms set limits to the number of connections to a broker that can be simultaneously started over TCP/IP, in accordance with the maximum value of the backlog size. Backlog is the buffer for connections in the TCP stack—the number of simultaneous TCP connection startups cannot exceed the backlog size. For example, Windows 2000 Professional limits the backlog to 5, and Windows 2000 Server limits the backlog to 200.
- You cannot edit a broker's instance configuration file without having started the broker instance at least once. This is because the `config.properties` file does not exist until the broker instance is first started. To configure a broker to use pluggable persistence or to set other configuration properties, run the broker once (with the instance name that should be used to create the broker) to create the `config.properties` file:

  `IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`
  (`/var/imq/instances/`*`brokerName`*`/props/config.properties` on Solaris)

  Once the `config.properties` file has been created, edit the file to add any configuration property values and then restart the broker.

Enterprise Edition Only

- The broker's shared thread pool model does not work on Windows platforms (due to a bug in J2SE 1.4.0). This is expected to be fixed in J2SE 1.4.1, but has not yet been tested.
- 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.
- 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.
- A connection service using SSL is currently limited to supporting only self-signed server certificates, that is, host-trusted mode. The connection configuration property `imqSSLIsHostTrusted` is set to true by default.

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

  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 "Resource in conflict" exception. This is not a real problem; it's 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.

Known Bugs

This section contains a listing of the more important bugs known at the time of the MQ 3.0.1 SP2 release.

For a list of current bugs, their status, and workarounds, Java Developer Connection (TM) 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 not all MQ bugs are listed here, it is a good starting place if you want to know whether a problem has been reported.

The relevant page is:

http://developer.java.sun.com/developer/bugParade

NOTE Java Developer Connection membership is free but does require registration for access. 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.

Table 3 Bug Descriptions

Bug Number	Details
4431924	`imqadmin`: modal dialogs can get into deadlock situation The Administration Console (`imqadmin`) uses dialogs that are application modal. Most of these dialogs are brought up explicitly by interacting with the graphical user interface, for example, by selecting the Add Brokers menu item. However, a dialog can also appear as a result of a lost broker connection. When more than one dialog is open, the Administration Console is locked. You will not be able to dismiss either modal dialog using the Close button. **Workaround**: Dismiss the top most dialog using the window manager controls i.e. - the 'X' button at the upper right corner on Windows - the 'Close' window manager menu item on Solaris
4449354	In extremely rare cases, calling the methods `Connection.stop`, `Connection.start`, and `Connection.close` at the same time as calling the methods `Session.recover` and `Session.rollback` (in separate threads) may result in an unexpected message redelivery order. **Workaround**: make sure your calls to the `Connection`… and `Session`… methods specified above are serialized either by using the same thread or by using synchronization.
4683029	The `-javahome` option in all solaris/win scripts does not work if the value has a space. The `-javahome` option is used by the MQ commands and utilities to specify an alternate Java 2 compatible runtime to use. However, the path to the alternate Java runtime must be located at a path that does not contain spaces. Examples of paths that have spaces are: Windows: `C:\jdk 1.4` (On Windows the path can contain spaces if the entire path is placed in quotes; for example, `"C:\jdk 1.4"`) Solaris: `/work/java 1.4` **Workaround**: Install the Java runtime at a location or path that does not contain spaces.
4689962	In the Japanese locale, the output of various admin utilities is not properly aligned in columns and the borders are too short. **Workaround**: None

Table 3 Bug Descriptions *(Continued)*

Bug Number	Details
4703406	QueueBrowser should function without first calling connection.start(). Connection.start() must be called on a Connection before a QueueBrowser can browse a Queue. If you fail to call Connection.start() the QueueBrowser enumeration will block on nextElement(), and eventually throw a java.util. NoSuchElementException. **Workaround**: Call Connection.start() before calling `QueueBrowser.getEnumeration()`.
4753010	Unbounded growth in Java process native heap segment with server VM. This is a Java VM bug that might affect the broker process under conditions of rapid opening and closing of client connections. It can cause continuous and unbounded broker process memory growth. The broker can eventually crash when all the system memory is exhausted. You can verify the growth of the native heap segment using the `/usr/proc/bin/pmap` tool. **Workaround**: Force the broker to use the client VM, using the following command option: `imqbrokerd -clientvm` or you can remove the `-server` argument to the `/usr/bin/imqbrokerd` shell script that starts the broker, or you can upgrade to JDK 1.4.1 and use the following command option: `imqbrokerd -javahome` *JDK_1.4.1_directory*
4761626	Heavy consumer creation/destruction w/ autocreate queues can cause message loss In rare stress situations it is possible for a message to be lost from an auto-created queue destination. The error will occur if, after all messages delivered from the queue have been acknowledged, the queue's last message consumer is removed at the same time a new message producer (on a different connection) sends a message to that queue. What happens is that the queue is "auto-removed" (it is empty without any consumers) just as the new message arrives, causing the new message to be lost. **Workaround**: Create queues administratively using MQ administration tools.
4879448	Sun ONE Message Queue Version 3.0.1 is not swapping persistent messages to disk when the heap fills to RED memory states. There is a boolean expression that determines if a message has incorrect logic. If the logic is incorrect, then the message should be swapped out. This means that in some cases, persistent messages will be swapped out of memory. If the ability to swap non-persistent messages is turned off it always evaluates correctly for persistent messages **Workaround**: Set the following broker property in either the `config.properties` file or through the command line: `imqmemory_management.swapNPMsps=false`

Table 3 Bug Descriptions *(Continued)*

Bug Number	Details
4883126	The Auto-reconnect feature does not work properly. The number of consumers grows after each auto-reconnect which causes the broker to send duplicate messages. **Workaround:** None. This bug will be fixed in MQ 3.5.
4888270	Re-transmitting a message originally sent in a transaction causes Broker Error If a message that was originally sent in a transaction is subsequently received and the same message object is re-transmitted in a non-transacted session, then the broker reports an error. **Workaround:** Do not use the same message object. Rather make a separate, new message when you really want to re-transmit the message.
4893546	Broker does not release memory when running w/jdk1.4.2 and server vm **Workaround:** Use the clientvm. The broker should be started using the following command: `imqbrokerd -clientvm`

Bugs Fixed in Message Queue 3.0.1 Versions

Below is a short description of the most important bugs fixed in MQ 3.0.1, 3.0.1 SP1, and 3.0.1 SP2.

For bugs fixed in MQ 3.0, see the MQ 3.0 *Release Notes* available at

http://docs.sun.com/?p=/coll/S1_MessageQueue_30

For more details about any of the following bug fixes you can view the complete report at the Java Developer Connection site:

http://developer.java.sun.com/developer/bugParade

Table 4 Bugs Fixed in MQ 3.0.1 versions

Bug Number	Description
4679837	Client sometimes throws `JMSException` on connection.`close()` when TLS is used as transport.
4683129	Broker log timestamp is incorrect between midnight and 1am.
4688051	Client can get an `EOFException` when rapidly creating and closing connections to the broker.

Table 4 Bugs Fixed in MQ 3.0.1 versions *(Continued)*

Bug Number	Description
4694971	Standalone JTS/XA transaction completion may fail occasionally, even after a normal, error-free XAConnection.close().
4700851	Client does not clean up local transaction after ending an XA transaction
4701982	The Administration Console cannot be launched on Solaris or Linux if the CLASSPATH environment variable is set
4702152	For messages acknowledged as part of a consumer-side transaction, broker holds on to some unnecessary state information for each acknowledged message until the consumer is closed.
4704186	Corrections in the example applications `README` file (`demo/jms/README`)
4735757	A consumer will throw an exception if more than 50 messages are received in a transaction prior to calling commit().
4758424	Performance degrades on round-robin queue when > 10k message backlog.
4758427	Round-robin queue consumers can get “stuck” if the last queued message expires prior to delivery.
4770212	Pausing and resuming a service causes client connections to hang if the connections have no pending messages.
4770518	Windows broker service terminates on logout.
4809079	The broker does not shut down properly if there are SSL based connections.
4821708	Session.createProducer(null) throws NullPointerException.
4828860	MQ performance degrades with selector - gets worse as # of clients increases.
4835586	Master broker backup - restore cycle can cause loss of destination information.
4879448	Under certain circumstances the broker will not swap messages to disk.

Functionality Marked as Optional in JMS

The JMS specification indicates certain items that are optional-- each JMS provider (vendor) chooses whether or not to implement them. The MQ product handling of each of these optional items is indicated below:

Table 5 Optional JMS Functionality

Section in JMS Specification	Description and MQ Handling
3.4.3 JMSMessageID	"Since message ID's take some effort to create and increase a message's size, some JMS providers may be able to optimize message overhead if they are given a hint that message ID is not used by an application. JMS Message Producer provides a hint to disable message ID."
	MQ implementation: Product does not disable Message ID generation (any setDisableMessageID() call in MessageProducer is ignored). All messages will contain a valid MessageID value.
3.4.12 Overriding Message Header Fields	"JMS does not define specifically how an administrator overrides these header field values. A JMS provider is not required to support this administrative option."
	MQ implementation: The MQ product supports administrative override of the values in message header fields through configuration of connection factory administered objects.
3.5.9 JMS Defined Properties	"JMS Reserves the 'JMSX' Property name prefix for JMS defined properties." "Unless noted otherwise, support for these properties is optional."
	MQ implementation: The JMSX properties defined by the JMS 1.1 specification are supported in the MQ product.
3.5.10 Provider-specific Properties	"JMS reserves the '`JMS_<vendor_name>`' property name prefix for provider-specific properties."
	MQ implementation: The purpose of the provider-specific properties is to provide special features needed to support JMS use with provider-native clients. They should not be used for JMS to JMS messaging. MQ 3.0.1 does not use provider-specific properties.
4.4.8 Distributed Transactions	"JMS does not require that a provider support distributed transactions."
	MQ implementation: Distributed transactions are supported in this release of the MQ product.
4.4.9 Multiple Sessions	"For PTP <point-to-point distribution model>, JMS does not specify the semantics of concurrent QueueReceivers for the same queue; however, JMS does not prohibit a provider from supporting this." See section 5.8 of the JMS specification for more information.
	MQ implementation: The MQ implementation supports three queue delivery policies: Failover, Round Robin, and Single (default). For more information, refer to the MQ *Administrator's Guide*.

Technical Notes

This section contains short write-ups on the following topics:

System Clock Settings

When using an MQ system, you should be careful to synchronize system clocks and avoid setting them backward.

Synchronization Recommended

It is recommended that you synchronize the clocks on all hosts interacting with the MQ system. This is particularly important if you are using message expiration (TimeToLive). Failure to synchronize the hosts' clocks may result in TimeToLive not working as expected (messages may not be delivered). You should synchronize clocks before starting any brokers.

Solaris You can issue the `rdate` command on a local host to synchronize with remote host. (You must be superuser--that is, root--to run this command.) For example, the following command synchronizes the local host (call it Host 2) with remote host Host1:

```
# rdate Host1
```

Linux The command is similar to Solaris, but you must provide the -s option:

```
# rdate -s Host1
```

Windows you can issue the net command with the time subcommand to synchronize your local host with a remote host. For example, the following command synchronizes the local host (call it Host 2) with remote host Host1:

```
net time \\Host1 /set
```

Avoid Setting System Clocks Backwards

You should avoid setting the system clock backwards on systems running an MQ broker. MQ uses timestamps to help identify internal objects such as transactions and durable subscriptions. If the system clock is set backwards it is theoretically possible that a duplicate internal identifier can be generated. The broker attempts to compensate for this by introducing some randomness to identifiers and by detecting clock shift when running, but if the system clock is shifted backwards by a significant amount when a broker is not running, then there is a slight risk of identifier duplication.

If you need to set the system clock backwards on a system running a broker by more than a few seconds, it is recommended that you either do it when there are no transactions or durable subscriptions, or do it when the broker is not running, then wait the amount of time you have shifted the clock before bringing the broker back up.

But the ideal approach is to synchronize clocks before starting up any brokers, and then use an appropriate technique to ensure that clocks don't drift significantly after deployment.

OS-Defined Connection Limitations on Clients and Brokers

On the Solaris and Linux platforms, the shell in which the client or broker is running places a soft limit on the number of file descriptors that a client can use. In the MQ system, each connection a client makes, or each connection a broker accepts, uses one of these file descriptors. As a result, you cannot have a broker or client running with more than 256 connections on Solaris or 1024 on Linux without changing this limit. (The number is actually slightly lower than that due to file descriptors that are used for other purposes, such as for file-based persistence.)

To change this limit, see the `ulimit` man page or the instructions under "Increasing File Descriptors to Improve File-based Persistence Performance," below. The limit needs to be changed in each shell in which a client or broker will be executing.

Increasing File Descriptors to Improve File-based Persistence Performance

On the Solaris and Linux platforms, the speed of storing messages in the default file-based persistence is affected by the number of file descriptors available for use by the file store. (Windows does not have a file descriptor limit.) A large number of descriptors will allow the system to process large numbers of persistent messages faster.

To improve performance for performance testing or deployment, administrators should increase the maximum number of file descriptors available to an application (in this case, the broker process) and then increase the size of the shared file descriptor pool used by the broker by updating the value of the property:

```
imq.persist.file.message.fdpool.limit
```

The value of this property must be less than the maximum number of file descriptors available on your system.

On Solaris, for example, you can increase the file descriptor limits using the `ulimit` command. Processes inherit system limits from their parent (login) shell. On Solaris, there is a “hard” limit and a “soft” limit. For a non-root user, the number of file descriptors for an application cannot exceed the soft limit, which, in turn, cannot exceed the hard limit.

To check the current file descriptor limits:

```
Hard limit: $ ulimit -Hn
Soft limit: $ ulimit -n
```

To change the file descriptor limits for “root” user:

```
# ulimit -Hn unlimited
# ulimit -n unlimited
```

After this, any process created from this shell will be able to open unlimited file descriptors. So it is safe to run the `imqbroker` command at this point.

To change the file descriptor limit for non-root user:

```
$ ulimit -Hn number1
$ ulimit -n number2
```

where *number1* is less than 1024, and *number2* is less than *number1*.

If 1024 is not enough, you have the following options:

- Run the broker as root.
- Write some "setuid" program to increase the ulimit value before running the broker. (Such programs pose tremendous security risk. Highly discouraged.)
- Tune the `rlim_fd_max` parameter in the `/etc/system` file and reboot the system.

Securing Persistent Data

The broker uses a persistent store that can contain, among other information, message files that are being temporarily stored. Since these messages might contain proprietary information, it is recommended that the data store be secured against unauthorized access.

A broker can use either the built-in or plugged-in persistence.

Built-in Persistent Store

A broker using built-in persistence writes persistent data to a flat file data store located at:

```
IMQ_VARHOME/instances/brokerName/filestore/
(/var/imq/instances/brokerName/filestore/ on Solaris)
```

where *brokerName* is a name identifying the broker instance.

The *brokerName*`/filestore/` directory is created when the broker instance is started for the first time. The procedure for securing this directory depends on the operating system on which the broker is running.

Solaris and Linux The permissions on the `IMQ_VARHOME/instances/`*brokerName*`/filestore/` directory depend on the umask of the user that started the broker instance. Hence, permission to start a broker instance and to read its persistent files can be restricted by appropriately setting the umask. Alternatively, an administrator (superuser) can secure persistent data by setting the permissions on the `IMQ_VARHOME/instances` directory to 700.

Windows The permissions on the `IMQ_VARHOME/instances/`*brokerName*`/filestore/` directory can be set using the mechanisms provided by the Windows operating system that you are using. This generally involves opening a properties dialog for the directory.

Plugged-in Persistent Store

A broker using plugged-in persistence writes persistent data to a JDBC Compliant database.

For a database managed by a database server (for example, an Oracle database), it is recommended that you create a user name and password to access the MQ database tables (tables whose names start with "IMQ"). If the database does not allow individual tables to be protected, create a dedicated database to be used only by MQ brokers. See the database vendor for documentation on how to create user name/password access.

The user name and password required to open a database connection by a broker can be provided as broker configuration properties. However it is more secure to provide them as command line options when starting up the broker (see MQ *Administrator's Guide*, Appendix A, "Setting Up Plugged-in Persistence").

For an embedded database that is accessed directly by the broker via the database's JDBC driver (for example, a Cloudscape database), security is usually provided by setting file permissions (as described in "Built-in Persistent Store," above) on the directory where the persistent data will be stored. To ensure that the database is readable and writable by both the broker and the `imqdbmgr` utility, however, both should be run by the same user.

Broker Memory Configuration

When the broker gets close to exhausting the JVM heap space used by Java objects, it uses various techniques such as flow control and message swapping to free memory. Under extreme circumstances it even closes client connections in order to free the memory and reduce the message inflow. Hence it is desirable to set the maximum JVM heap space high enough to avoid such circumstances.

However, if the maximum Java heap space is set too high, in relation to system physical memory, the broker can continue to grow the Java heap space until the entire system runs out of memory. This can result in diminished performance, unpredictable broker crashes, and/or affect the behavior of other applications and services running on the system.

This problem can be avoided by configuring a sensible Java heap size limit using the `-Xmx` Java command line argument. In general it is a good idea to evaluate the normal and peak system memory footprints, and configure the Java heap size so that it is large enough to provide good performance, but not so large as to risk system memory problems.

Client Out of Memory Errors

If you are running a JMS client that deals with large messages or many small messages, it may encounter `OutOfMemoryError` errors. The client runtime does not have a memory leak--it just has insufficient memory to copy the messages off the network and deliver them to your client.

To eliminate these `OutofMemoryError` errors, increase the maximum Java heap size. You can do this by passing the appropriate command line option to the `java` or `jre` command.

On Java2, use the `-Xmx` option when running the client application. For example:

```
java -Xmx128m MyClass
```

Please note these limitations:

- The maximum limit of the VM's memory allocation pool (heap size) depends on both the operating system and the JDK release. Please check the JDK documentation for restrictions.
- The size of the VM's memory allocation pool must be less than or equal to the amount of virtual memory available on the system.

How to Report Problems

To report a problem, send mail to imq-feedback@sun.com.

If you have a support contract and you have problems with MQ, contact Sun ONE customer support using one of the following mechanisms:

- Sun ONE online support web site at http://www.sun.com/service/sunone/software/index.html

  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

For More Information

Beyond the MQ documentation, you can find additional information as indicated below.

Discussion Forums

Sun ONE Software Forum

There is a Sun ONE MQ forum available at the following location:

http://swforum.sun.com/jive/forum.jsp?forum=24

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. Email your comments to Sun at this address:

docfeedback@sun.com

Please include the part number (817-3731-10) of the document in the subject line and the book title (*Message Queue 3.0.1 Release Notes*) in the body of your email.

Additional Sun Resources

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

- Documentation for Sun ONE Message Queue
  http://docs.sun.com/coll/S1_MessageQueue_301
- Sun ONE Message Queue product information
  http://sun.com/software/message_queue
- Sun ONE Documentation
  http://docs.sun.com/prod/sunone
- Sun ONE Professional Services
  http://www.sun.com/service/sunps/sunone
- Sun ONE Software Products and Service
  http://www.sun.com/software
- Sun ONE Software Support Services
  http://www.sun.com/service/sunone/software
- Sun ONE Support and Knowledge Base
  http://www.sun.com/service/support/software
- Sun Support and Training Services
  http://www.sun.com/supportraining
- Sun ONE Consulting and Professional Services
  http://www.sun.com/service/sunps/sunone
- Sun ONE Developer Information
  http://sunonedev.sun.com
- Sun Developer Support Services
  http://www.sun.com/developers/support
- Sun ONE Software Training
  http://www.sun.com/software/training
- Sun Software Data Sheets
  http://wwws.sun.com/software