Sun Java™ System Message Queue Release Notes

Version 3.5 SP1

Part Number 817-6022-10

These release notes contain important information available at the time of release of version 3.5 SP1 of Sun Java™ System Message Queue (formerly Sun™ ONE Message Queue). The document also includes the contents of the Message Queue 3.5 release notes for customers who are upgrading from pre-3.5 versions. New features and enhancements, known limitations and problems, technical notes, and other information about Message Queue 3.5 versions are addressed here.

The most up-to-date version of these release notes can be found at the Sun Java System documentation web site: `http://docs.sun.com/coll/MessageQueue_35_SP1`. Check the web site before 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
March 12, 2004	Updated bug information. Updated “Known Issues and Limitations” section. Added “Redistributable Files” section. Updated “Documentation Updates” section. Updated “Compatibility Issues” section. Updated “Sun Java System Information” section.
January 9, 2004	Updated information for PointBase 4.8 version support; updated information for C-API features.

About Message Queue 3.5 SP1

Message Queue 3.5 SP1 is an update of Message Queue 3.5 and includes all the new features of Message Queue 3.5. In addition, Message Queue 3.5 SP1 includes bug fixes and a new brand name. The product now belongs to the Sun Java™ System family of products.

Message Queue 3.5 SP1 has been certified as compliant with the Java™ Message Service (JMS) 1.1 specification: it has passed the JMS 1.1 Compatibility Test Suite (CTS).

This section describes the changes made in Message Queue 3.5 SP1 and the changes made in the previous release, Message Queue 3.5.

Message Queue 3.5 SP1

Message Queue 3.5 SP1 contains bug fixes and rebranding of the product and the documentation.

Message Queue 3.5

Message Queue 3.5 included many new features:

These are described in the following sub-sections.

C Client Support (Enterprise Edition)

Message Queue 3.5 includes a C API and C runtime support (referred to hereafter as the C client feature). The C client feature can be used to integrate legacy systems into a Message Queue messaging system. It is an almost-full implementation of the JMS specification. It supports all JMS functions except: certain body types (map, stream, and object), queue browser functions, and J2EE application server functionality (such as distributed transactions and `ConnectionConsumer` objects).

Support for the C client feature is provided by a separately-installed set of libraries that is enabled only with an Enterprise Edition license. Hence, an upgrade from the Platform Edition to the Enterprise Edition requires installing both the Enterprise Edition license file and the C libraries.

Platform Edition customers who enable the trial, 90-day Enterprise Edition license can use the C client feature if they contact Sun using the `imq-feedback@sun.com` alias, requesting the C-API SDK. Engineering will be responsible for responding to these requests and making the C-API SDK available on the anonymous FTP site. After the 90-day Enterprise Edition license expires, customers can continue to build C clients, but they cannot connect them to their Platform Edition broker.

The C client feature requires specific compiler versions on the different operating system platforms, adding new system requirements to the Enterprise Edition (see the *Message Queue Installation Guide* for details). The C client feature also has dependencies on the Netscape Portable Runtime (NSPR) and Network Security Service (NSS) libraries. (In Message Queue 3.5, the C client feature was tested successfully on Linux Red Hat Advanced Server 2.1. The NSPR and NSS library versions it was tested against have not been certified for that Linux edition.)

Currently the C-API does not support the `basic` authentication type. If you configure the broker to use this authentication type, a call to the `MQCreateConnection` function will fail with the result `MQ_UNSUPPORTED_AUTH_TYPE`.

Documentation of the C client feature includes both reference documentation, programming documentation, and example C-API clients. For more information, see the *Message Queue C Client Developer's Guide*.

Java Client Connection Failover (Enterprise Edition)

Message Queue 3.5 supports an enhanced auto-reconnect capability by which a failed connection can be restored not only on the original broker, but also on a different broker (client connection failover). The reconnect is to the message service rather than to a specific broker instance. To implement this behavior, you configure the connection factory administered object (Message Queue 3.5 has a new message service address specification scheme), specifying a set of broker addresses (`imqAddressList`). When the client runtime needs to establish (or re-establish) a connection to a message service, it will attempt to connect to the brokers in the list in a prioritized order, until it finds an available broker or fails to find one. You can specify the number of connection attempts (`imqAddressListIterations`) on each of these brokers, and the interval between connection attempts (`imqAddressListInterval`).

If the auto-reconnect is to a broker instance different from the original, persistent messages and other state information held by the failed (or disconnected) broker can be lost. This is because the various broker instances in a cluster do not use a shared, highly available persistent store. However, the ability of the client runtime to automatically reconnect to a different broker instance allows you to create recovery scenarios by which a backup broker or a broker cluster can be used for (less than complete) failover protection.

Also, if auto-reconnect is enabled, Message Queue 3.5 will now persist *temporary* destinations when the associated connection fails, due to the possibility that clients might re-connect and access them again. Temporary destinations will be treated like other physical destinations; this might require you to routinely purge a broker of any unused temporary destinations.

For more information, see the *Message Queue Java Client Developer's Guide*.

Message Queue previously supported an auto-reconnect capability, by which the client runtime can automatically reconnect to a broker if a connection fails, except in situations where the client-side state cannot be fully restored on the broker upon reconnect (for example, when using transacted sessions or temporary destinations, which exist only for the duration of a connection).

Enhanced Broker Message Flow Control

Enhancements have been introduced in the broker to better control the flow of messages into destinations and to avoid situations in which the production of messages was much faster than their consumption. (In addition, other new Message Queue 3.5 features might help eliminate bottlenecks in the flow of messages *out* of destinations. See "Enhanced Java Client Flow Control" on page 6 and "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8.)

The broker's message flow enhancements include:

- Placing a limit on the number of producers associated with a destination. Destinations now have a new `maxNumProducers` attribute; when this limit is reached, no new producers can be created for the destination.
- Enabling new, configurable limits and limit behaviors when destinations reach `maxTotalMsgBytes` and `maxNumMsgs` limits. In particular, the following changes have been implemented:
    - Extending the `maxTotalMsgBytes` and `maxNumMsgs` destination attributes to topic destinations (they previously applied only to queue destinations)
    - Enabling the setting of `maxTotalMsgBytes` and `maxNumMsgs` for *auto-created* destinations
    - (Enterprise Edition) Allowing an administrator to choose a behavior in response to reaching any of the above limits. You can specify the following behaviors: slowing producers (`FLOW_CONTROL`), throwing out oldest messages (`REMOVE_OLDEST`), throwing out lowest priority messages according to age of the messages (`REMOVE_LOW_PRIORITY`), and/or throwing out the newest message (`REJECT_NEWEST`).
    - Implementing the producer flow control limit behavior (`FLOW_CONTROL`) on a *per-producer*, rather than a per-connection basis. (The previous implementation shut down all producers on a connection when too many messages had been received by a destination through the connection.) Per-producer flow control shuts down only those producers on a connection associated with the flooded destination, allowing other producers on the connection to continue to send messages to other destinations.

- Allowing an administrator to pause (and resume) a specific destination. You can pause the delivery of messages from producers to the destination, or from the destination to consumers, or both. This is done through two new `imqcmd` subcommands: `pause` and `resume`, as illustrated below:
    - `imqcmd pause dst -n myQueue -t q -pst PRODUCERS`
    - `imqcmd resume dst -n myQueue -t q`

For more information, see the *Message Queue Administration Guide*.

Enhanced Java Client Flow Control

The Message Queue 3.5 client runtime manages the flow of messages on a *per-consumer*, as well as a per connection basis. You can place a limit on the number of messages buffered per consumer, thereby preventing any one consumer from being overwhelmed by other consumers. This capability also means that in the case of queue delivery to multiple consumers, you can better balance delivery of messages among multiple consumers. It also helps you manage memory resources in the Message Queue client runtime.

A new connection factory attribute, `imqConsumerFlowLimit`, limits the number of messages buffered *per consumer* for all consumers sharing a common connection. When the number of messages in a consumer buffer drops below a threshold percentage (`imqConsumerFlowThreshold`) of `imqConsumerFlowLimit`, the broker can deliver another batch of messages to the client runtime for consumption by that consumer. If the total number of messages buffered for all consumers on a connection exceeds the `imqConnectionFlowLimit`, then delivery of messages through the connection will stop until that total drops below the connection limit.

(The previous implementation of client runtime flow control let you place a limit on the number of messages buffered in the client runtime, waiting to be consumed (`imqConnectionFlowLimit`). The purpose of this feature was to limit the amount of client memory used for buffering messages so that slowly consuming clients would not crash from running out of memory. This feature was implemented at the connection level; that means that if a connection supports many consumers, an avalanche of messages for one consumer could prevent other consumers from receiving messages.)

For more information, see the *Message Queue Java Client Developer's Guide*.

New Destination Metrics

Message Queue 3.5 includes enhanced tracking of messages and consumers per destination to allow for better monitoring and control of memory and usage.

The new metrics appear as output of the new `imqcmd metrics dst` subcommand. This command displays cumulative totals (since the initiation of sampling), current values, average values (calculated over samples taken), and peak values (since the initiation of sampling) for both message and consumer metrics.

For example, the `imqcmd metrics dst -m ttl` command returns the following information:

- Message flow
    - Message Flow In: cumulative total, rate
    - Message Flow Out: cumulative total, rate
- Messages stored in broker (unacknowledged messages in memory or in persistent store)
    - Number of Messages: current, peak, average
    - Message Bytes: current, peak, average
- Largest Message so far, Bytes

The `imqcmd metrics dst -m con` command returns the following information:

- Number of Active Consumers: current, peak, average (see "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8)
- Number of Backup Consumers: current, peak, average (see"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8)

For more information, see the *Message Queue Administration Guide*.

Remote Monitoring API (Enterprise Edition)

Message Queue 3.5 provides a message-based API by which remote (or local) JMS clients can easily monitor and analyze broker metrics. The API is based upon the broker being able to produce messages that contain metrics information about the broker itself, the Java VM, and individual destinations (see "New Destination Metrics" on page 6). These messages are sent to specific topic destinations, depending on the entity being monitored, whenever one or more consumers is subscribed to these destinations. The consuming client can then retrieve the messages, use a header property (`type`) to filter them, and then extract the metrics information they contain.

For more information, see both the *Message Queue Administration Guide* and the *Message Queue Java Client Developer's Guide*.

(Message Queue previously only supported local logging of broker metrics data and remote queries of metrics information using Message Queue administration utilities. These features, while providing important metrics data, did not facilitate easy analysis of this data.)

Message Queue Resource Adapter for JMS (J2EE Application Server Support)

Message Queue 3.5 includes a JMS Resource Adapter for plugging the Message Queue JMS message service into any compliant J2EE application server.

A resource adapter is a standardized way for plugging additional functionality into a J2EE application server (by connecting to an EIS, a messaging system, and so forth), in compliance with the J2EE Connector Architecture Specification (JCA 1.5). This architecture allows any J2EE application server, for example, to support JMS messaging by connecting to any JMS provider that implements JCA 1.5: J2EE components deployed and running in the application server environment can exchange JMS messages using the plugged-in JMS provider (client runtime and server).

For more information, see the *Message Queue Administration Guide*.

Custom Message Acknowledgement

Message Queue presently supports the JMS `CLIENT_ACKNOWLEDGE` client acknowledgement mode, by which a JMS client explicitly acknowledges consumption of messages. In `CLIENT_ACKNOWLEDGE` mode, the client invokes the `acknowledge()` method of a message object, causing the session to acknowledge all messages that have been consumed by the session since the previous invocation of the method.

Message Queue 3.5 enhances this behavior by letting you acknowledge *individual* messages. That is, you can acknowledge only a specific message, rather than acknowledge, as a batch, all the messages consumed up to that time. This is achieved in code by casting the message object to a special Message Queue message type upon which you invoke a new `acknowledge()` method. This allows you to deviate from the JMS standard to handle special application needs.

For more information, see the *Message Queue Java Client Developer's Guide*.

Enhanced Queue Delivery Policies (Enterprise Edition)

The implementation of queue delivery to multiple consumers, previously implemented as three distinct queue delivery policies (single, failover, and round-robin), has changed. Message Queue 3.5 uses a more generalized approach in which delivery is load balanced among a configurable number of active (and backup) consumers. The Message Queue 3.5 implementation is based on the following new destination attributes:

- `maxNumActiveConsumers`: Specifies the number of consumers—one or many—active in load-balanced queue delivery
- `maxNumBackupConsumers`: Specifies the number of backup consumers—none or many—that can take the place of active consumers should any active consumers fail.

(New consumers will be rejected if the number of consumers exceeds the sum of these two attributes.)

The Message Queue Platform Edition supports load-balanced queue delivery for up to two consumers, and the Enterprise Edition supports an unlimited number of consumers.

The new load balancing mechanism takes into account the message consumption rate of different consumers. It works like this:

- An initial number of queued messages in a destination are routed to available active consumers (in the order in which they registered with the destination) in batches of a configurable size (the destination's `consumerFlowLimit` attribute). Once these messages have been delivered, additional messages arriving in a destination are routed to consumers one-by-one, as consumers become available (that is, as consumers acknowledge all messages previously delivered to them). If an active consumer fails, then the first backup consumer is made active, and takes over the work of the failed consumer.
- In a broker cluster environment, the delivery mechanism can be set to prioritize local consumers. A new destination attribute, `localDeliveryPreferred`, lets you specify that messages be delivered to remote consumers only if there are no consumers on the local broker—the broker on which the destination was created. This lets you increase performance in situations where routing to remote clients (through their respective home brokers) might cause slowdowns in throughput. (This attribute requires that the destination's scope not be restricted to local-only delivery—see "Enhanced Cluster Performance (Enterprise Edition)" on page 9.)

For more information, see the *Message Queue Administration Guide*.

Enhanced Cluster Performance (Enterprise Edition)

In a broker cluster environment, destinations are replicated on all brokers, and all messages delivered to those destinations are forwarded to all brokers that have consumers registered for those destinations, even if only a small percentage of the messages will be delivered to any given consumer (for example, as in the case of a durable subscriber using selection criteria, or a queue receiver involved in load-balanced queue delivery). This broker-to-broker traffic can cause message avalanches, especially when a new consumer becomes active. To reduce excessive broker-to-broker traffic in a cluster, Message Queue 3.5 introduces the following enhancements:

- Adopting new flow-control mechanisms that regulate the delivery of messages to a consumer connection. In other words, the *consumer* will regulate the delivery of messages (from destination to client runtime), avoiding the passing of unnecessary messages from broker to broker. (These mechanisms should also help prevent logjams on the client runtime side—see "Enhanced Java Client Flow Control" on page 6).
- Changing the implementation queue delivery to multiple consumers (see "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8) to cut down on unnecessary passing of messages from broker to broker. This implementation includes a new queue destination attribute, `localDeliveryPreferred`, which lets you specify that local consumers get priority over remote consumers in queue delivery to multiple consumers (see "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8).

For more information, see the *Message Queue Administration Guide*.

Local Destinations (Enterprise Edition)

A new destination attribute, `isLocalOnly`, lets you specify a destination as limited to delivering messages only to local consumers (consumers connected to the broker on which the destination is created) as opposed to consumers connected to other brokers in a cluster. Similarly, it can only receive messages sent to it from local producers. You can use this property to create independent, non-interacting destinations of the same name on different brokers within a cluster, and to set up failover-like scenarios by which a message is sent to two destinations in case one of them should fail.

For more information, see the *Message Queue Administration Guide*.

Secure Broker Clusters (Enterprise Edition)

In situations in which secure, encrypted message delivery between client and message server is required, Message Queue 3.5 now supports secure delivery of messages between the brokers in a cluster. To achieve secure, encrypted delivery of messages within a cluster, you have to configure the internal cluster connection service to use an SSL-based transport protocol.

For more information, see the *Message Queue Administration Guide*.

Enhanced Persistent Store Performance

The implementation of both the Message Queue flat-file data store and the JDBC-compliant data store has been changed in Message Queue 3.5 to improve performance. These enhancements are described in the following two sections. For more information, see the *Message Queue Administration Guide*.

Built-in Persistence (Flat-file Data Store)

The performance enhancement in the Message Queue flat-file data store involves internal changes in the data format that only surfaces in upgrading from Message Queue 3.01 (or earlier) to Message Queue 3.5.

The migration of the file store is handled automatically when a Message Queue 3.5 broker instance is initially started up and references an earlier version of file store. A copy of the earlier file store is saved in the instance directory, and you will need to delete it manually once the migration has been completed. To remove the earlier file store automatically—in the case where there is not enough disc space for two copies of the store—you can start the version 3.5 broker with an added option, as shown in the following command:

```
imqbrokerd -upgrade-store-nobackup
```

(There are no spaces in `upgrade-store-nobackup`.)

The root of the new flat-file data store has been changed from:

```
…/instances/instanceName/filestore/
```

to:

```
…/instances/instanceName/fs350/.
```

In addition, the Message Queue Command utility (`imqcmd`) has been enhanced to provide file store metrics:

```
imqcmd metrics dst -n destName -t type -m dsk
```

And `imqcmd` will include a new compacting command:

```
imqcmd compact dst -n destName -t type
```

Plugged-in Persistence (JDBC™-Compliant Data Store)

The Message Queue JDBC-compliant data store has been changed in Message Queue 3.5 to support broker memory management enhancements (see "Enhanced Broker Message Flow Control" on page 5 and "New Destination Metrics" on page 6) and a broader range of database vendors. The modifications to support enhanced memory management (categorizing messages by destination) are schema changes that are handled transparently, but support for additional database vendors involves changes in how you configure plugged-in persistence.

The migration of the JDBC Compliant store is handled automatically when a Message Queue 3.5 broker instance is initially started up with an earlier version of the persistent store. The old tables will be kept intact, however, and will need to be deleted manually using a new `imqdbmgr delete oldtbl` command once the migration is completed. To remove the earlier tables automatically--in the case where there is not enough space for two copies of the store--you can initially start the broker with a new option: `imqbrokerd -upgrade-store-nobackup`.

As for the additional database support, you could not previously customize SQL statements to the JDBC Compliant database you want to plug in (the SQL statements were internally generated). Message Queue 3.5 now includes new instance configuration properties that enable you to customize the SQL code that creates the Message Queue database schema. There is a configurable property for each database table: the property is the SQL code that creates the table. These properties are needed to properly specify the data types used by the plugged-in database. Examples are provided based on a PointBase embedded database, rather than the earlier Cloudscape database.

Instance-Specific Authentication and Authorization

By default, Message Queue 3.5 provides for each instance to have its own file-based user repository and its own access control file, both placed in a standard location: .../`instances/`*instanceName*`/etc/`. These two files will be created when a broker instance is started for the first time. If the broker finds these files in the old location (typical in an upgrade from a previous version), it will copy the files to the instance-specific location. If the broker does not find these files in the old location (typical in a new installation), it will place default versions of the files in the instance-specific location.

To support instance-specific user repositories, a -i *instanceName* option has been added to the User Manager utility (`imqusermgr`) to specify the instance-specific user repository to which each `imqusermgr` command applies.

For more information, see the *Message Queue Administration Guide*.

(Previously, by default, all instances of a broker on a single computer share the same file-based user repository (and hence the same client login password) and the same access control file. However, you could configure each broker instance to use a specific LDAP user repository location or to use a specific access control file, both specified in the instance configuration file.)

RPM-Based Linux Installation

Message Queue 3.5 installation on Linux is performed using the Red Hat Package Manager (RPM), a command line driven package management system capable of installing, uninstalling, verifying, querying, and updating software packages (RPMs).

In addition, the installed directory structure for Message Queue on Linux has changed to match standard locations used for unbundled products on Linux. (Both Solaris™ and Linux platforms have standards that depend on whether a product is bundled or unbundled with the operating system.) In particular, there is no longer a root Message Queue installation directory on Linux, similar to the Solaris situation.

For more information, see the *Message Queue Installation Guide*.

Support for Solaris Operating System, X86 Platform Edition

On Solaris 9, Message Queue 3.5 is supported for X86 processors in addition to SPARC processors.

Hardware and Software Requirements

Hardware and software required for this release, and supported products and platforms are described in detail in the *Message Queue Installation Guide*.

Bugs Fixed

This section includes short descriptions of fixed bugs, as follows:

- Table Table 2 on page 13 describes bugs fixed in Message Queue 3.5 SP1.
- Table 3 on page 14 describes bugs fixed in Message Queue 3.5.

For older lists of bug fixes, see the following:

- For Message Queue 3.0.1 Service Pack 2, see the *Message Queue 3.0.1 Service Pack 2 Release Notes* at:

  `http://docs.sun.com/coll/S1_MessageQueue_301_SP2`

- For Message Queue 3.0.1, see the *Message Queue 3.0.1 Release Notes* at:

  `http://docs.sun.com/coll/S1_MessageQueue_301`

- For Message Queue 3.0, see the *Message Queue 3.0 Release Notes* at:

  `http://docs.sun.com/coll/S1_MessageQueue_30`

For more details about a bug fix, you can view the complete report at the Java Developer Connection site:

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

Fixed in Message Queue 3.5 SP1

Table 2 lists and describes the bugs fixed in Message Queue 3.5 SP1. (Table 3 on page 14 lists and describes the bugs fixed in Message Queue 3.5.)

Table 2 Bugs Fixed in Message Queue 3.5 SP1

Bug Number	Description
4942723	Broker may run out of memory sending large messages with the shared thread pool option.
4944894	Broker may occasionally generate `CancelledKeyException` when using shared thread pool.
4947239	Creating and closing producers repeatedly causes a small amount of client memory growth.
4947993	Can not destroy destination or durable with an active durable subscriber.
4948525	Negative numbers may be shown in metrics output for Message Bytes In and out. This will occur when more than 2143510810 bytes have been sent.

Table 2 Bugs Fixed in Message Queue 3.5 SP1 *(Continued) (Continued)*

Bug Number	Description
4948563	Packet conversion: INFO message displayed on each 2.0 SP1 msg sent to 3.5 broker. Each time a 2.0 SP1 client sends a message to a 3.5 broker, the following INFO level message is displayed: `[04/Nov/2003:10:34:16 PST] Internal Error: Unknown ProducerUID 0`
4949781	Unable to use cluster broadcaster error while starting Broker.
4952332	Messages may be delivered out of order if the primary consumer fails over to a backup consumer on the same connection.
4956748	Cannot use a master broker with the Oracle database.
4964703	C-API: The `MESSAGE_ID` header returned by the `MQGetMEssageHeaders()` function is not prefixed with “ID:”
4964712	C-API: The `MESSAGE_ID` header set by the `MQSetMEssageHeaders()` function is not ignored on message send.
4969583	C-API: The same message handle should be able to call `MQAcknowledgeMessages()` more than once.
4983150	The `JMSRedlivered` flag is not set when a broker is restarted and redelivers the message.
4983699	Broker loses exceptions thrown by the store when it fails to persist a message.

Fixed in Message Queue 3.5

Table 3 lists and describes the bugs fixed in Message Queue 3.5.

Table 3 Bugs Fixed in Message Queue 3.5

Bug Number	Description
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.
4630183	Destroying a destination leaves durable subscriptions on the broker
4753010	Unbounded growth in Java process native heap segment with server VM.
4761626	Heavy consumer creation/destruction w/ autocreate queues can cause message loss
4855307	Broker can not authenticate against an LDAP repository because default configuration uses old property name (`bindDN`)
4883126	The Auto-reconnect feature does not work properly.
4888270	Re-transmitting a message originally sent in a transaction causes Broker Error

Table 3 Bugs Fixed in Message Queue 3.5 *(Continued)*

Bug Number	Description
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.
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`.
4866814	On solaris, Broker can not log error and warning messages using syslog if it has been started with a 64 bit jvm (the broker is started with '-vmargs -d64'). This occurs because the beta release of Message Queue does not contain a 64 bit version of our library, `libimqutil.so.1`.
4872121	The broker does not start on a non-networked system which does not have an IP address other than 127.0.0.1.
4879902	Slow memory increase in broker.
4881968	New monitoring clients can not be created if `imq.autocreate.topic` is set to false.
4884827	CTS1.3 MDB/EJB CMT tests fail with Message Queue 3.5 & AppServer 7.0
4885654	Producers may fail if a new message is published to an autocreated destination at the same time the system is reaping the destination.
4887506	During failover from a single primary consumer to a backup consumer, messages may be delivered out of order.
4888939	C and Java clients on a destination with a behavior of FLOW_CONTROL may stop receiving messages if the maximum size of a destination (`maxNumMsgs`) is very small (< 5 messages).
4889002	Property imq.transaction.autorollback is not supported in 3.5beta
4891874	Consumer-based flow control may cause messages to stop being delivered to consumers. This problem is more likely to occur with 4896133: ConnectionConsumers and the Sun Java System Application Server 7's Message Driven Beans.
4895262	HTTPS clients failed to connect to the broker through `HTTPSTunnelServlet`
4897500	In a cluster, when a client calls unsubscribe() to remove a durable subscription, it is only removed from the broker that the client is connected to. This means that messages produced to other brokers will continue to be stored for that subscriber.
4898020	Message Queue 3.0.* and Message Queue 3.5 brokers cannot be used together in a cluster. Starting a mixed cluster will generate a error on the 3.0.1 broker: Configuration mismatch: Aborting connection with broker [...] because following configuration properties do not match - null imq.queue.deliverypolicy

Table 3 Bugs Fixed in Message Queue 3.5 *(Continued)*

Bug Number	Description
4888983	`imqcmd list dur` does not display durable subscribers with the same durable name

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
- Compatibility Issues
- Documentation Updates

Installation Notes

Refer to the *Message Queue Installation Guide* for information about system requirements, supported software platforms and products, pre-installation instructions, upgrade procedures, and all other information relevant to installing Message Queue on the Solaris, Linux, and Windows platforms.

Compatibility Issues

This section covers compatibility issues in Message Queue 3.5 SP1 and in Message Queue 3.5.

Issues Related to the Next Major Release of Message Queue

The following are incompatible changes that might be introduced in the next major release of Message Queue. This information is provided now in order to allow you to prepare for these changes.

- Message Queue client support for all releases of J2SE 1.3 will be dropped. J2SE 1.4 will continue to be supported.

- All Message Queue command line interfaces will be modified to remove the option of providing a password as a command line argument. For example:

  ```
  imqbrokerd -ldappassword <passwd> imqcmd -p <passwd>
  ```

  Alternative mechanisms to specify the password will be provided.
- The format of the broker's log file will change. Applications that depend on the current format might no longer work.
- The locations of individual files installed as part of Message Queue might change. This could break existing applications that depend on the current location of certain Message Queue files.
- The `imqkeytool` program might be removed from the product. The J2SE keytool will be the supported replacement.
- Message Queue clients that use a version of Message Queue older than the next major version might not have access to the new features offered in that version of the product.
- No error is generated when the `MQAcknowledgeMessages() f`unction is called by C clients (using `MQ_CLIENT_ACKNOWLEDGE`) on a message that has already been acknowledged. This behavior might change.

Issues In Message Queue 3.5

Message Queue 3.5 is generally compatible with Message Queue 3.0 (and subsequent versions 3.0.1, 3.0.1 Service Pack 1, and 3.0.1 Service Pack 2). However, changes have been made in broker properties, administered objects, persistence schema, file locations, and administration tools that can impact an upgrade from Message Queue 3.0 versions to Message Queue 3.5.

The Message Queue 3.5 install operation does not remove or over-write the Message Queue 3.0 `IMQ_VARHOME` directory. This directory contains configuration and security-related files. Most of this data is compatible with Message Queue 3.5, and can be preserved using the instructions in the *Message Queue Installation Guide*.

The issues that you might need to address when upgrading from Message Queue 3.0 to Message Queue 3.5 include the following:

- Broker Compatibility
- Property and Attribute Changes
- Location of Public .jar Files

For information about administered object compatibility, client compatibility, and administration tool compatibility, please see the *Message Queue Installation Guide*.

Broker Compatibility

An Message Queue 3.5 broker will inter-operate with an Message Queue 3.0 broker, however changes have been made in broker properties and the persistent store schema. Some Message Queue 3.0 data is still compatible with Message Queue 3.5. For further information, see the *Message Queue Installation Guide*.

Property and Attribute Changes

This section contains a summary of changes in broker properties, destination attributes, and connection factory attributes made in Message Queue 3.5.

Broker Properties The following tables detail new properties, deprecated properties, and property name changes in Message Queue 3.5. For details, see Chapter 2 of the *Message Queue Administration Guide.*

Table 4 New Message Queue 3.5 Broker Properties

Property Name	Feature Reference
`imq.persist.file.message.max_record.size`	"Enhanced Persistent Store Performance" on page 10
`imq.persist.file.destination.message.filepool.limit`	"Enhanced Persistent Store Performance" on page 10
`imq.metrics.topic.enabled`	"Remote Monitoring API (Enterprise Edition)" on page 7
`imq.metrics.topic.interval`	"Remote Monitoring API (Enterprise Edition)" on page 7
`imq.metrics.topic.persist`	"Remote Monitoring API (Enterprise Edition)" on page 7
`imq.metrics.topic.timetolive`	"Remote Monitoring API (Enterprise Edition)" on page 7
`imq.autocreate.destination.maxNumMsgs`	"Enhanced Broker Message Flow Control" on page 5
`imq.autocreate.destination.maxTotalMsgBytes`	"Enhanced Broker Message Flow Control" on page 5
`imq.autocreate.destination.maxBytesPerMsg`	"Enhanced Broker Message Flow Control" on page 5
`imq.autocreate.destination.maxNumProducers`	"Enhanced Broker Message Flow Control" on page 5
`imq.autocreate.queue.maxNumActiveConsumers`	"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8
`imq.autocreate.queue.maxNumBackupConsumers`	"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8

Table 4 New Message Queue 3.5 Broker Properties *(Continued)*

Property Name	Feature Reference
`imq.autocreate.queue.consumerFlowLimit`	"Enhanced Java Client Flow Control" on page 6 and "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8
`imq.autocreate.topic.consumerFlowLimit`	"Enhanced Java Client Flow Control" on page 6
`imq.autocreate.queue.localDeliveryPreferred`	"Enhanced Cluster Performance (Enterprise Edition)" on page 9
`imq.autocreate.destination.isLocalOnly`	"Local Destinations (Enterprise Edition)" on page 10

Table 5 Deprecated Broker Properties in Message Queue 3.5

Property Name
`imq.persist.file.message.fdpool.limit`
`imq.persist.file.message.filepool.limit`
`imq.redelivered.optimization`
`imq.queue.deliverypolicy`

The following tables detail new and deprecated destination attributes in Message Queue 3.5. For details, see Chapter 6 of the *Message Queue Administration Guide*.

Table 6 New Message Queue 3.5 Destination Attributes

Destination Type	Attribute Name	Feature Reference
Queue & Topic	`maxNumMsgs`	"Enhanced Broker Message Flow Control" on page 5
Queue & Topic	`maxTotalMsgBytes`	"Enhanced Broker Message Flow Control" on page 5
Queue & Topic	`limitBehavior`	"Enhanced Broker Message Flow Control" on page 5
Queue & Topic	`maxBytesPerMsg`	"Enhanced Broker Message Flow Control" on page 5
Queue & Topic	`maxNumProducers`	"Enhanced Broker Message Flow Control" on page 5
Queue only	`maxNumActiveConsumers`	"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8
Queue only	`maxNumBackupConsumers`	"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8

Table 6 New Message Queue 3.5 Destination Attributes *(Continued)*

Destination Type	Attribute Name	Feature Reference
Queue & Topic	`consumerFlowLimit`	"Enhanced Java Client Flow Control" on page 6 and "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 8
Queue only	`localDeliveryPreferred`	"Enhanced Cluster Performance (Enterprise Edition)" on page 9
Queue & Topic	`isLocalOnly`	"Local Destinations (Enterprise Edition)" on page 10

Table 7 Deprecated Destination Attributes in Message Queue 3.5

Destination Type	Attribute Name
Queue	`QueueDeliveryPolicy`

Connection Factory Attributes The following table details new connection factory attributes in Message Queue 3.5. For details, see Chapter 4 of the *Message Queue Java Client Developer's Guide*. Note that Message Queue 3.0 version connection factory attributes are still supported in Message Queue 3.5, and will continue to be supported until the next major release of the Message Queue product.

Table 8 New Message Queue 3.5 Connection Factory Attributes

Attribute Name	Feature Reference
`imqAddressList`	"Java Client Connection Failover (Enterprise Edition)" on page 4
`imqAddressListBehavior`	"Java Client Connection Failover (Enterprise Edition)" on page 4
`imqAddressListIterations`	"Java Client Connection Failover (Enterprise Edition)" on page 4
`imqReconnectEnabled`	"Java Client Connection Failover (Enterprise Edition)" on page 4
`imqReconnectAttempts`	"Java Client Connection Failover (Enterprise Edition)" on page 4
`imqReconnectInterval`	"Java Client Connection Failover (Enterprise Edition)" on page 4
`imqConsumerFlowLimit`	"Enhanced Java Client Flow Control" on page 6
`imqConsumerFlowThreshold`	"Enhanced Java Client Flow Control" on page 6

Table 9 Renamed Connection Factory Attributes

Previous Name	Message Queue 3.5 Name
`imqFlowControlCount`	`imqConnectionFlowCount`
`imqFlowControlIsLimited`	`imqConnectionFlowLimitEnabled`
`imqFlowControlLimit`	`imqConnectionFlowLimit`

Location of Public .jar Files

On the Solaris platform, the location of public `.jar` files was moved in Message Queue 3.0.1 from their Message Queue 3.0 location (`/usr/share/lib/imq/`) to the following location: `/usr/share/lib/`. The symbolic links that were introduced in Message Queue 3.0.1 in `/usr/share/lib/imq/` for the jar files that were moved, have been removed.

This applies to the following `.jar` files:

- `jms.jar`
- `imq.jar`
- `imqxm.jar`
- `activation.jar`
- `saaj-api.jar`
- `saaj-impl.jar`
- `mail.jar`
- `commons-logging.jar`
- `jaxm-api.jar`
- `fscontext.jar`

Documentation Updates

This section describes the changes made to Message Queue documentation in version 3.5 SP1 and in version 3.5.

Changes in Version 3.5 SP1

The following Message Queue 3.5 SP1 documents were updated from Version 3.5 of the product:

Installation Guide

The *Message Queue Installation Guide* was updated to reflect branding changes and platform support information.

Administration Guide

The *Message Queue Administration Guide* was renamed (from *Message Queue Administrator's Guide*) and has been updated to reflect branding changes. This document also contains an expanded, updated chapter describing performance monitoring.

Java Client Developer's Guide

The *Message Queue Java Client Developer's Guide* was updated to reflect branding changes.

C Client Developer's Guide

The *Message Queue C Client Developer's Guide* was updated to reflect branding changes.

Changes in Version 3.5

The following Message Queue 3.5 documents were updated from Version 3.0.1 of the product. These updated documents can be found at the Message Queue 3.5 documentation web site: `http://docs.sun.com/coll/S1_MessageQueue_35`.

Installation Guide

The Message Queue 3.5 product includes an updated *Message Queue Installation Guide*. This includes new software requirements, changes in Solaris installation instructions, a new Linux installation procedure using Red Hat Package Manager (RPM) and installed directory structure, and minor changes in the Windows installation.

Correction: For plugged-in persistence support, Table 1-2 specifies that PointBase Version 4.5 is supported by Message Queue. In fact, the version supported is PointBase, Version 4.8.

Administrator's Guide

The *Message Queue Administration Guide* has been updated to include changes in Message Queue 3.5 (see "This section describes the changes made in Message Queue 3.5 SP1 and the changes made in the previous release, Message Queue 3.5." on page 2).

Java Client Developer's Guide

The *Message Queue Java Client Developer's Guide* includes most of what was contained in the previous Message Queue *Developer's Guide*, and has been updated to include changes in Message Queue 3.5 (see "This section describes the changes made in Message Queue 3.5 SP1 and the changes made in the previous release, Message Queue 3.5." on page 2).

C Client Developer's Guide

The *Message Queue C Client Developer's Guide* is a new book that has been added to the Message Queue documentation set to describe how to create Message Queue C client applications.

Known Issues and Limitations

This section describes known issues, limitations, and bugs in Message Queue 3.5 SP1 and Message Queue 3.5. Because version 3.5 SP1 is simply a rebranded version of Message Queue 3.5, the information presented in this section applies to both versions.

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.

The relevant page is:

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

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`.

Known Issues

This section covers known issues in Message Queue 3.5 SP1. Some of these were introduced with the Message Queue 3.5 version. This section groups issues according to whether they apply to both Enterprise and Platform Editions of Message Queue 3.5 or to the Enterprise Edition only.

Both Enterprise and Platform Editions

- Due to product rebranding, APIs which previously returned the string:

  ```
  "Sun ONE Message Queue, Sun Microsystems, Inc."
  ```

  will now return the string:

  ```
  "Sun Java(tm) System Message Queue"
  ```

- Due to product rebranding, C client programs compiled with MQ 3.5 FCS that do an exact comparison of the value associated with the `MQ_NAME_PROPERTY` from `MQGetMetaData()` will fail when using the 3.5 SP1 `mqcrt` shared library at runtime.

- 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.

- If you are running Windows XP, there is a limit to the number of *inbound* connections. For Windows XP Professional, the maximum number of other computers that are permitted to simultaneously connect over the network is ten. This limit includes all transports and resource sharing protocols combined. For Windows XP Home Edition, the maximum number of other computers that are permitted to simultaneously connect over the network is five. This limitation will affect the number of clients that can connect to the broker running Windows XP.

  Any file, print, named pipe, or mail slot session that does not have any activity is automatically disconnected after the `AutoDisconnect` time has expired; the default for the `AutoDisconnect` time is 15 minutes. When the session is disconnected, one of the ten connections becomes available so that another user can connect to the Windows XP system. Therefore, lowering the `AutoDisconnect` time can help reduce some of the problems with the ten-connection limit or the five-connection limit on a system that is not used heavily for server purposes. For more information, see the following:

  http://support.microsoft.com/default.aspx?scid=kb;EN-US;314882

- 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:

Platform	Location
Solaris	`/var/imq/instances/`*instanceName*`/props/config.properties`
Linux	`/var/opt/imq/instances/`*instanceName*`/props/config.properties`
Windows	`IMQ_VARHOME\instances\`*instanceName*`\props\config.properties`

  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

- 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.
- 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 "Client ID is already in use" 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

Table 10 lists bugs outstanding in Message Queue 3.5 SP1.

Table 10 Known Bugs in Message Queue 3.5

Bug Number	Details
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 Message Queue 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.
4939923	Broker may generate `NullPointerException` when using shared thread pool and the broker’s JVM is running low on memory. **Workaround**: None. This bug is fixed in J2SE 1.4.2_03.
4941058	Destinations with flow control on may not reach the max limit. There are situations where producers may be stopped from sending messages to a destination prior to the destination reaching its configured max limit. **Workaround**: None
4941066	Destinations can go slightly over assigned byte limits. **Workaround**: None
4941127	Destination will not completely load if a message exceeds individual message size limit. If the limit on the size of messages allowed in a destination is changed after a larger message is stored, the destination will not load cleanly. **Workaround**: Increase the message size limit until the large message is consumed, then lower the message limit. One could pause producing to the destination during this time to prevent other large messages from being accepted.
4946531	Innocuous `NullPointerException` may rarely occur while producing messages. **Workaround**: None - the null pointer exception can be safely ignored.

Table 10 Known Bugs in Message Queue 3.5 *(Continued)*

Bug Number	Details
4949398	`imqcmd query dst` reports incorrect values for `Number of Messages` and `Total Message Bytes` when the corresponding destination is being loaded. The values reported are correct before and after the destination is loaded. **Workaround**: The problem only occurs while the destination is being loaded. Once the destination is loaded, the values returned are correct.
4950166	Random errors in broker when running on jdk1.4.2_02 and x86 systems. For more information, see J2SE bug 4947404. **Workaround**: Start broker with `-XX:UseSSE=0`, for example `imqbrokerd -tty -vmargs -XX:UseSSE=0`

4950601	`imqcmd metrics dst` triggers broker internal error to be printed when using the JDBC persistent store.
	
	Disk usage metrics information applies to file store only. However, when metrics information is retrieved, the broker will try to get the disk usage information regardless of the store type. If a database is used instead of the file store, the following error message will be printed by the broker:
	
	`06/Nov/2003:22:57:36 PST] ERROR [B3100]: Unexpected Broker Internal`
	
	`Error : [unable to disk usage for destinationT:topic1] :`
	
	`com.sun.messaging.jmq.jmsserver.util.BrokerException:`
	`The operation does not apply to plugged-in persistent store.`
	
	Workaround: None

4951010	In a broker cluster, a broker will queue messages to a remote connection which may not be started. **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.
4953348	HTTPS `createQueueConnection` occasionally throws exception on Windows 2000. **Workaround**: Retry the connection.
4953354	Broker becomes inaccessible when persistent store opens too many destinations. **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.
4954974	Using the CD media, installation does not automatically start on Windows XP. **Workaround**: In Windows Explorer, double-click on the windows folder on the CD and then double-click on the `imq3_5-ent-win.exe` file to launch the installer.
4983525	Creating a message producer for an autocreated destination may fail on Linux Red Hat Advanced Server 3.0. **Workaround**: Attempt to recreate the producer. It should succeed the second time. Alternately use an administratively created destination.

Table 10 Known Bugs in Message Queue 3.5 *(Continued)*

Bug Number	Details

4986318	Client may unexpectedly generate ACKNOWLEDGE_REPLY message:
	`******** Packet: ACKNOWLEDGE_REPLY(25):26-192.18.86.227-42976-1075458056557`
	`Magic/Version: 469754818/301Size: 97 Type: ACKNOWLEDGE_REPLY(25)`
	`Expiration: 0       Timestamp: 1075458056557`
	`Source IP: 192.18.86.227  Source Port: 42976Sequence: 26`
	Workaround: None. There is a rare timing condition in the broker that causes the client to generate this error. The error can be ignored. No messages are lost.

4991257	Sending large persistent messages to durable subscribers in a broker cluster where the persistent store is JDBC based may cause the broker to hang and/or generate errors. **Workaround**: Increase the broker's lock protocol timeout using the following broker property: `imq.cluster.timeout=<timeout-in-seconds>` The default is 60. If persisting large messages is slow, you might need to tune the persistent store database or switch to a different persistent store.

5006686	The ARGS example in imqbrokerd.conf is incorrect.
	Workaround: The values should not be quoted as in the example.
	`ARGS="-name newbroker -port 8888"`
	The value should be as follows:
	`ARGS=-name newbroker -port 8888`

Redistributable Files

Sun Java System Message Queue 3.5 SP1 contains the following set of files which you may use and freely distribute in binary form:

```
jms.jar
imq.jar
imqxm.jar
fscontext.jar
providerutil.jar
jndi.jar
ldap.jar
```

```
ldapbpjar
jaas.jar
jsse.jar
jnet.jar
jcert.jar
```

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

How to Report Problems and Provide Feedback

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

If you have a support contract and you have problems with Message Queue, contact 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 Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. Use the web-based form to provide feedback to Sun:

`http://www.sun.com/hwdocs/feedback/`

Please provide the full document title and part number in the appropriate fields. The part number can be found on the title page of the book or at the top of the document, and is usually a seven or nine digit number. For example, the part number of these *Message Queue 3.5 SP1 Release Notes* is 817-6022-10.

Additional Sun Resources

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

Discussion Forums

Sun Java System Software Forum

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

`http://softwareforum.sun.com/NASApp/jive/forum.jsp?forum=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`

SunSolve Knowledge Base

Information on Sun Java System Message Queue is available on line in the SunSolve Knowledge Base, located at:

`http://sunsolve.Sun.COM/pub-cgi/search.pl?mode=advanced`

Select "All Free Collections" and then search for "Message Queue".

Sun Java System Information

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

- Message Queue Product Page
  `http://wwws.sun.com/software/products/message_queue/index.html`
- Documentation for Message Queue
  `http://docs.sun.com/coll/MessageQueue_35_SP1`
- Sun Documentation
  `http://docs.sun.com/`
- Sun Java System Software Products and Services
  `http://www.sun.com/software`
- Sun Software Support Services
  `http://www.sun.com/service/sunone/software`
- Sun Support and Knowledge Base
  `http://sunsolve.sun.com`
- Sun Support and Training Services
  `http://www.sun.com/supportraining`
- Sun Developer Information
  `http://developers.sun.com/`
- Sun Developer Support Services
  `http://www.sun.com/developers/support`
- Sun Software Data Sheets
  `http://www.sun.com/software`