Sun ONE Message Queue Release Notes

Version 3.5

Part Number 817-3726-11

January 2004

These release notes contain important information available at the time of release of Version 3.5 of Sun™ Open Network 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 Message Queue 3.5.

The most up-to-date version of these release notes can be found at the Sun ONE documentation web site: `http://docs.sun.com/coll/S1_MessageQueue_35` 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
- About MQ Version 3.5
- What's New in MQ Version 3.5
- Bugs Fixed in MQ 3.5
- Hardware and Software Requirements
- Installation Notes
- Errata and MQ Documentation Updates
- Compatibility Issues
- Known Issues
- How to Report Problems and Provide Feedback
- Additional Sun Resources

Revision History

Table 1 Revision History

Date	Description of Changes
January 9, 2004	Updated information for PointBase 4.8 version support; updated information for C-API features.
November 25, 2003	MQ 3.5 release version

About MQ Version 3.5

For information about new and improved features of MQ 3.5, please see "What's New in MQ Version 3.5."

MQ 3.5 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).

The C client feature has been tested successfully on Linux Red Hat Advanced Server 2.1 (though the NSPR and NSS library versions it was tested against have not been certified for that Linux edition). Java clients have not been tested on JDK 1.2.

What's New in MQ Version 3.5

This section describes new features and feature enhancements included in MQ 3.5. These include the following:

"C Client Support (Enterprise Edition)" on page 3

"Java Client Connection Failover (Enterprise Edition)" on page 4

"Enhanced Broker Message Flow Control" on page 5

"Enhanced Java Client Flow Control" on page 6

"New Destination Metrics" on page 6

"Remote Monitoring API (Enterprise Edition)" on page 7

"MQ Resource Adapter for JMS (J2EE Application Server Support)" on page 8

"Custom Message Acknowledgement" on page 8

"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 9

"Enhanced Cluster Performance (Enterprise Edition)" on page 10

"Local Destinations (Enterprise Edition)" on page 10

"Secure Broker Clusters (Enterprise Edition)" on page 11

"Enhanced Persistent Store Performance" on page 11

"Instance-Specific Authentication and Authorization" on page 12

"RPM-Based Linux Installation" on page 13

"Support for Solaris Operating System, X86 Platform Edition" on page 13

C Client Support (Enterprise Edition)

MQ 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 an MQ 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 MQ *Installation Guide* for details). The C client feature also has dependencies on the Netscape Portable Runtime (NSPR) and Network Security Service (NSS) libraries.

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 MQ *C Client Developer's Guide*.

Java Client Connection Failover (Enterprise Edition)

MQ 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 (MQ 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, MQ 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 MQ *Java Client Developer's Guide*.

MQ 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 MQ 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 9.)

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 MQ *Administrator's Guide*.

Enhanced Java Client Flow Control

The MQ 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 MQ 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 MQ *Java Client Developer's Guide*.

New Destination Metrics

MQ 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 9)
- Number of Backup Consumers: current, peak, average (see"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 9)

For more information, see the MQ *Administrator's Guide*.

Remote Monitoring API (Enterprise Edition)

MQ 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 MQ *Administrator's Guide* and the MQ *Java Client Developer's Guide*.

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

MQ Resource Adapter for JMS (J2EE Application Server Support)

MQ 3.5 includes a JMS Resource Adapter for plugging the MQ 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 MQ *Administrator's Guide*.

Custom Message Acknowledgement

MQ 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.

MQ 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 MQ 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 MQ *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. MQ 3.5 uses a more generalized approach in which delivery is load balanced among a configurable number of active (and backup) consumers. The MQ 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 MQ 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 10.)

For more information, see the MQ *Administrator's 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, MQ 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 9) 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 9).

For more information, see the MQ *Administrator's 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 MQ *Administrator's Guide*.

Secure Broker Clusters (Enterprise Edition)

In situations in which secure, encrypted message delivery between client and message server is required, MQ 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 MQ *Administrator's Guide*.

Enhanced Persistent Store Performance

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

Built-in Persistence (Flat-file Data Store)

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

The migration of the file store is handled automatically when a MQ 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 MQ 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 MQ JDBC-compliant data store has been changed in MQ 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 MQ 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). MQ 3.5 now includes new instance configuration properties that enable you to customize the SQL code that creates the MQ 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, MQ 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 MQ *Administrator's 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

MQ 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 (RPM's).

In addition, the installed directory structure for MQ 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 MQ installation directory on Linux, similar to the Solaris situation.

For more information, see the MQ *Installation Guide*.

Support for Solaris Operating System, X86 Platform Edition

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

Bugs Fixed in MQ 3.5

Table 2 on page 14 provides short descriptions of the most important bugs fixed in MQ 3.5.

- For bugs fixed in MQ 3.0.1 Service Pack 2, see the MQ 3.0.1 Service Pack 2 *Release Notes* available at

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

- For bugs fixed in MQ 3.0.1, see the MQ 3.0.1 *Release Notes* available at

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

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

  `http://docs.sun.com/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 2 Bugs Fixed in MQ 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
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 MQ 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 MQ 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).

Table 2 Bugs Fixed in MQ 3.5 *(Continued)*

Bug Number	Description
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 ONE 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	MQ3.0.* and MQ3.5 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
4888983	`imqcmd list dur` does not display durable subscribers with the same durable name

Hardware and Software Requirements

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

Installation Notes

Please refer to the MQ *Installation Guide* for information about system requirements, supported software platforms and products, pre-installation instructions, upgrade procedures, and all other information relevant to installing MQ 3.5 on the Solaris, Linux, and Windows platforms.

Errata and MQ Documentation Updates

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

Installation Guide

The MQ 3.5 product includes an updated MQ *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 MQ. In fact, the version supported is PointBase, Version 4.8.

Administrator's Guide

The MQ *Administrator's Guide* has been updated to include changes in MQ 3.5 (see "What's New in MQ Version 3.5" on page 2).

Java Client Developer's Guide

The MQ *Java Client Developer's Guide* includes most of what was contained in the previous MQ *Developer's Guide*, and has been updated to include changes in MQ 3.5 (see "What's New in MQ Version 3.5" on page 2).

C Client Developer's Guide

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

Compatibility Issues

MQ 3.5 is generally compatible with MQ 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 MQ 3.0 versions to MQ 3.5.

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

The issues that you might need to address when upgrading from MQ 3.0 to MQ 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 MQ *Installation Guide*.

Broker Compatibility

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

Property and Attribute Changes

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

Broker Properties

The following tables detail new properties, deprecated properties, and property name changes in MQ 3.5. For details, see Chapter 2 of the MQ *Administrator's Guide*.

Table 3 New MQ 3.5 Broker Properties

Property Name	Feature Reference
`imq.persist.file.message.max_record.size`	"Enhanced Persistent Store Performance" on page 11
`imq.persist.file.destination.message.filepool.limit`	"Enhanced Persistent Store Performance" on page 11
`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 9
`imq.autocreate.queue.maxNumBackupConsumers`	"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 9
`imq.autocreate.queue.consumerFlowLimit`	"Enhanced Java Client Flow Control" on page 6 and "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 9
`imq.autocreate.topic.consumerFlowLimit`	"Enhanced Java Client Flow Control" on page 6
`imq.autocreate.queue.localDeliveryPreferred`	"Enhanced Cluster Performance (Enterprise Edition)" on page 10
`imq.autocreate.destination.isLocalOnly`	"Local Destinations (Enterprise Edition)" on page 10

Table 4 Deprecated Broker Properties in MQ 3.5

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

Destination Attributes

The following tables detail new and deprecated destination attributes in MQ 3.5. For details, see Chapter 6 of the MQ *Administrator's Guide*.

Table 5 New MQ 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 9
Queue only	`maxNumBackupConsumers`	"Enhanced Queue Delivery Policies (Enterprise Edition)" on page 9
Queue & Topic	`consumerFlowLimit`	"Enhanced Java Client Flow Control" on page 6 and "Enhanced Queue Delivery Policies (Enterprise Edition)" on page 9
Queue only	`localDeliveryPreferred`	"Enhanced Cluster Performance (Enterprise Edition)" on page 10
Queue & Topic	`isLocalOnly`	"Local Destinations (Enterprise Edition)" on page 10

Table 6 Deprecated Destination Attributes in MQ 3.5

Destination Type	Attribute Name
Queue	QueueDeliveryPolicy

Connection Factory Attributes

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

Table 7 New MQ 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 8 Renamed Connection Factory Attributes

Previous Name	MQ 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 MQ 3.0.1 from their MQ 3.0 location (`/usr/share/lib/imq/`) to the following location: `/usr/share/lib/.` The symbolic links that were introduced in MQ 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`

Known Issues

This section contains a description of known limitations, including the most important remaining open defects.

Known Limitations

Limitations shown in this section are grouped according to whether they apply to both Enterprise and Platform Editions of MQ 3.5 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.
- 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

This section contains a listing of the more important bugs known at the time of the MQ 3.5 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 9 Descriptions of Known Bugs

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 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.
4895667	See the description of bug number 4949781.
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.
4942723	Broker may run out of memory sending large messages with the shared thread pool option. **Workaround**: 1. Modify the JVM's direct byte buffer limit: See the workaround in bug 4879883 to set `-XX:MaxDirectMemorySize=XXX` where `XXX` is the applicable size e.g. 1024m 2. Override the broker to use direct buffers: `Dimq.packet.read.override=""`

Table 9 Descriptions of Known Bugs *(Continued)*

Bug Number	Details
4944894	Broker may occasionally generate `CancelledKeyException` when using shared thread pool. **Workaround**: When running with the -shared thread pool on a multi-cpu system, the broker may periodically display `cancelledKeyException` Depending on the state of the connection when this problem occurs it may either cause no errors or cause the connection to be closed. The client can reconnect to the broker and continue to send messages.
4946531	Innocuous `NullPointerException` may rarely occur while producing messages. **Workaround**: None - the null pointer exception can be safely ignored.
4947239	Creating and closing producers repeatedly causes a small amount of client memory growth. **Workaround**: Closing the connection frees the memory.
4947993	Can not destroy destination or durable with an active durable subscriber. **Workaround**: A workaround exists. For more information see bug 4947993.
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. **Workaround**: None
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` **Workaround**: None. While the messages will bloat the broker's log file they will not affect compatibility between old clients and 3.5 brokers.
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.
4949781	Unable to use cluster broadcaster error while starting Broker. This is a well-known Linux installation issue. The Red Hat installer always adds an entry in the `/etc/hosts` file that maps the host name to the loopback interface address (127.0.0.1). This has been documented as a known bug in the J2SE 1.4.2 release notes: `http://java.sun.com/j2se/1.4.2/docs/guide/net/relnotes.html` Please see BugID 4665037 for more details. **Workaround**: If the host has a static IP address, the simple workaround is to correct `/etc/hosts` so that the host name resolves to the correct address. Alternatively, if the address is static and is registered in the name service (DNS for example), then change `/etc/nsswitch.conf` so that the name service is consulted prior to `/etc/hosts`.

Table 9 Descriptions of Known Bugs *(Continued)*

Bug Number	Details
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.
4952332	Messages may be delivered out of order if the primary consumer fails over to a backup consumer on the same connection. **Workaround**:Stop the connection before closing the consumer.
4952918	An `UnsupportedClassVersionError` is generated starting the broker with an older Java runtime. **Workaround**: Only start the broker with a supported Java Runtime: 1.4.2_02
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.

Table 9 Descriptions of Known Bugs *(Continued)*

Bug Number	Details
4954051	MQ Java client using transacted sessions w/JDK1.3 needs additional `javax` classes. When an MQ 3.5 Java Client that uses a transacted session is run with JDK1.3, a `NoClassDefFoundError` is thrown because the runtime cannot find certain javax.transaction.xa classes. **Workaround**: Download the `javax.transaction` Class files which are available online at: `http://java.sun.com/products/jta` Select 'Class Files 1.0.1B' and include the classes in the runtime classpath. or Use J2SE 1.4 to run the MQ 3.5 Java Client.
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.
4956748	Cannot use a master broker with the Oracle database. **Workaround**: Configure the master broker to use the file-based persistent store. This is only necessary for the master broker. Other brokers in the cluster can use the Oracle JDBC-based persistent store.

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 MQ, contact Sun ONE 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 Release Notes* is 817-3726-11.

Additional Sun Resources

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

Discussion Forums

Sun ONE Software Forum

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

http://softwareforum.sun.com/NASApp/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

SunSolve Knowledge Base

Information on Sun ONE 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 ″Sun ONE Message Queue".

Sun ONE Information

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

- Documentation for Message Queue
  http://docs.sun.com/coll/S1_MessageQueue_35
- Sun ONE Documentation
  http://docs.sun.com/prod/sunone
- Message Queue Product Page
  http://www.sun.com/software/message_queue
- 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

Copyright © 2004 Sun Microsystems, Inc. All rights reserved.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties.Sun, Sun Microsystems, the Sun logo, Java, Solaris and Sun[tm] ONE are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Copyright © 2004 Sun Microsystems, Inc. Tous droits réservés.

L'utilisation est soumise aux termes du contrat de licence.Cette distribution peut comprendre des composants développés par des tierces parties.Sun, Sun Microsystems, le logo Sun, Java, Solaris et Sun[tm] ONE sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays.UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exlusivement par X/Open Company, Ltd.