Java™ Message Queue Deployment Guide

V1.1

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part Number: 806-4795-10
Revision A, May 2000

Java Message Queue Deployment Guide

Please
Recycle

May 2000

Contents

Preface

The Java Message Queue Deployment Guide provides information related to deploying your Java Message Queue™ (JMQ) applications.

Who Should Use This Book

The primary audience for this book is the developer deploying Java Message Queue applications.

Before You Read This Book

This manual is a member of the Java Message Queue document set. Each document addresses a different aspect of the JMQ product.

You should be familiar with the following two documents before reading this Guide:

- INSTALL
- *Java Message Queue Quickstart Guide*

For information on the installation of the Java Message Queue product and on platform-specific issues, see INSTALL. For information on running JMQ applications, including common errors, refer to the *Java Message Queue Quickstart Guide* and the *Java Message Queue Deployment Guide*.

You can also refer to the Javadoc™ for the Java Message Service and to the Java Message Service specification documentation.

How This Book Is Organized

This book is organized into the following chapters:

Chapter 1, "Introduction to JMQ Systems," describes how to deploy JMQ applications and provides detailed information on the environment variables that you need for to run JMQ systems.

Chapter 2, "Command-Line Options," describes command options for the JMQ router, JMQ applications, and the JMQ utilities `irmon` and `ircmd`.

Chapter 3, "JMQ Router and Supporting Utilities," describes options and commands for the JMQ Router and JMQ utilities—`irmon` and `ircmd`.

Chapter 4, "Configuring JMQ Administered Objects," describes using the `jmqconfig` utility to configure JMQ Administered Objects.

Chapter 5, "JMQ Administration Console," describes the `jmqadmin` utility, the use of which can assist you in the administration of the Java Message Queue system.

Chapter 6, "JMQ Router as a Windows Service," describes how to configure, and start the Windows Service and how to use jmqsvcadmin, which you can use to either install the JMQ Router as an Windows Service after your initial installation, or to remove the JMQ Router as an Windows Service after your initial installation.

Chapter 7, "Troubleshooting JMQ," discusses common problems that can occur when using JMQ applications.

Chapter 8, "Error Codes,"describes valid JMQ error codes and error strings.

Chapter 9, "System Monitoring Output," explains JMQ system print output and output levels.

A glossary is provided at the end of the book.

This manual describes how to deploy Java Message Queue applications, including setting the Java Message Queue environment and the router network. This manual also discusses common problems encountered, including run time errors.

Chapter 1: Introduction to JMQ Systems

This chapter provides introductory deployment information on Java Message Queue systems.

Before reading this chapter, you should be familiar with the concepts discussed in the Quickstart Guide. Of particular importance are the concepts of Java Message Queue routers, Java Message Queue applications, subnets, and Java Message Queue network topologies.

Running the JMQ System in Networks

Running the JMQ system in networks involves the following simple deployment steps:

1. Start the JMQ router.
2. Start the JMQ applications. JMQ applications must be connected to a router, which can be on any machine. If you do not specify the machine to connect to, the application attempts to connect to a router on the local machine.

   If there is no router running on the local machine, you need to start the JMQ application by specifying the remote machine to which to connect; the remote machine must be running a router.

   Note that JMQ applications do not connect directly together to communicate; rather, they communicate through one or more routers.
3. Create the router network. If you want to run more than one router, start the other routers using command-line options to connect the routers to each other. In their default configuration, the routers are not connected to each other. You can establish up to one router per machine per subnet (see "Subnets" below). You must govern your router placement by performance criteria such as bandwidth conservation over slow-speed links.You should begin your router placement by deploying just one router

Environment Variables

JMQ uses the environment variables described in this chapter to determine where to look for:

- Configuration files and program libraries

- Operational limits on runtime resource usage.

Environment Variables Table

The following table lists the Java Message Queue environment variables:

Environment Variable	Description
JMQ_HOME	Identifies the main directory in which JMQ is installed. You must define this variable for the JMQ router, utilities, and examples. On the Solaris™ platform, if JMQ_HOME is not explicitly set the default JMQ_HOME directory is: /opt/SUNWjmq On the Windows platform, the installation program sets JMQ_HOME to the installation directory by default, which is: C:\Program Files\JavaMessageQueue1.1
JMQ_CHARSET	Identifies the system encoding for characters. The valid values for this variable are ASCII, JISEUC, JISX, SJIS, KSC, BIG5, CWS, and UTF8. When this is not set, it is assumed to be ASCII.
JMQ_RESPATH	Read by the JMQ router to locate resource files on a host. The first directory is referred to as the "system directory" and the last directory is referred to as the "user directory." Directories are separated by a delimiter (see the example). At least one resource directory must be defined for the JMQ router and utilities. This variable is assumed to be the res subdirectory under the JMQ home directory on Windows. You should set this environment variable only to change the default behavior of resource management—to specify different directories for SYS or USER or both, and additional paths to search for resources. Default settings for JMQ_RESPATH are: In the Solaris™ operating environment: /var/opt/SUNWjmq/res/sys:/var/opt/SUNWjmq/res/user In the Windows operating environment: %JMQ_HOME%\res\sys;%JMQ_HOME%\res\user For more information on JMQ_RESPATH, see "Resource File Search" in this chapter.

Environment Variable	Description
JMQ_MAX_QUEUESIZE	The limit, in bytes, of the data file that will be held in router memory per connection for a JMQ client application. The default value assumed for this variable is 5,000,000 bytes. JMQ_MAX_QUEUESIZE and JMQ_MAX_FILESIZE are advanced features you can use to limit the size of data queued for a client that is slow to respond to messages. You typically do not need to change these values from their default. JMQ_MAX_QUEUESIZE limits the amount of data in memory that is waiting to be sent to a client on a single JMQ connection. This data can include messages for both topics and queues registered on that connection. Because of these limits JMQ will react in the following manner to clients that are too slow in processing messages sent to them: ■ First, store the client's message data in router memory. ■ Second, if router memory (JMQ_MAX_QUEUESIZE) becomes too large, cache the message data in a disk file. ■ If the disk file exceeds the maximum designated size, the JMQ router breaks the connection to the client application. You need to adjust these values only if the JMQ system has clients that are processing messages much slower than the JMQ router is sending messages. An administrator might need to decrease JMQ_MAX_QUEUESIZE if the router is running out of swap space. When JMQ_MAX_QUEUESIZE is decreased, non-responsive clients use less memory. An administrator might want to raise JMQ_MAX_QUEUESIZE if the JMQ router system has sufficient memory, and speed is critical (because file access is slower than memory access).

Environment Variable	Description

JMQ_MAX_FILESIZE	The limit, in bytes, of the data file in which the router will cache the data that is being held by the router per connection for a JMQ client application. The default value assumed for this variable is 20,000,000 bytes.
	
	Each connection can cache to disk up to 20,000,000 bytes of data that is waiting to be sent to a client before failing the connection.
	
	JMQ_MAX_FILESIZE limits the size for the cached queue. If the cached queue exceeds JMQ_MAX_FILESIZE, the router closes the JMS connection related to the cached queue, freeing up the data queued to the client both in memory and on disk..
	
	You might want to decrease JMQ_MAX_FILESIZE to limit the amount of disk space needed for each client and/or to force the router to close the connection sooner to a client that is slow in processing data.
	
	You might want to increase JMQ_MAX_FILESIZE if disk space is not at a premium. Increasing this value enables more data to be queued for a slow client before the router fails the connection.
	
	If you have a JMQ client application whose connection is frequently closed by the router, and you want to debug this action, you can restart the router with the -ds1 debug option.
	
	If the router prints the following messages just before it closes the connection, the value of JMQ_MAX_FILESIZE is probably too small. Increasing the value should prevent this from occurring again:
	`Maximum channel disk buffer size exceeded...closing channel.`
	`Refer to the JMQ_MAX_FILESIZE environment variable in the Deployment Guide for additional help.`
	
	`Channel closed...`

Subnets

A router subnet defines a JMQ communications channel between all routers and applications that need to talk to each other.

Subnet Restrictions

Routers and applications must be on the same subnet to communicate with each other. JMQ applications that use different subnets do not "see" each other; routers on different subnets also do not communicate. Because applications and routers on different subnets are not aware of each other, multiple JMQ systems can coexist on just a few hosts.

If you want your JMQ applications to communicate, the applications must use the same subnet. If you do not specify a subnet for your JMQ applications or routers to use, JMQ uses subnet 0.

Subnet Number Assignment

A JMQ application uses a predictable port number when it connects to another JMQ application. The port that JMQ uses for TCP is (9312_{10} + *n*) where *n* is the JMQ subnet number (specified by the `-n` option). Subnet 10 (specified by `-n10`) thus translates to port 9322. Valid subnet numbers are 0 to 999.

You can specify an absolute port number to use by passing a negative number to the `-n` option. For example, if you specify `-n-10`, JMQ uses port number 10. Valid port numbers are from 0 to 65535.

You must avoid using subnets that conflict with the ports that other software uses. Other software might possibly be impacted by JMQ packets. The non-JMQ packets might significantly degrade the performance of your JMQ applications; they would cause the link to be shut down and re-opened each time a non-JMQ packet is received.

If another software package is using a particular port, you cannot run a router on the corresponding subnet on the same host at the same time that the server for the other software package is running. The router will exit with an error message

```
]ERROR![ Subnet is in use, unable to proceed
```

You can receive this message if you start two routers on the same subnet.

TCP ports in the range of 0 to 1024 are reserved for super-user privilege on Solaris. To use a port in this range on the Solaris platform, you must run the JMQ router as `root`.

The Internet Assigned Numbers Authority (IANA) assigns port numbers used for standard protocols. To date, most assigned port numbers are in the range 1024 to 7200. The IANA has also assigned ports 9535 and 17007 to `man` and `isode-dua`.

You can obtain an up-to-date listing of the assigned port numbers by entering the following URL into your web browser `ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers`. If you are using a UNIX machine, you should also refer to the `/etc/services` file on the UNIX machine for possible port conflicts.

Starting the JMQ Router

You can use special JMQ command-line options to configure how the JMQ router behaves once it is started. This section describes use of the router command-line options.

You can also tailor the startup configuration of JMQ applications you develop. However, you need to be aware of any options specific to the JMQ system.

For example, you might want to use the -s option as an option for your JMQ application; however, JMQ uses the -s option to specify the host to which to connect.

To solve such potential conflicts, you should therefore choose another command-line option for your JMQ application. In addition, JMQ interprets only command-line options after a double dash (`--`) as options. JMQ ignores any options your JMQ application would interpret before the double dashes.

Specifying Command-Line Options When Running `irouter`, `ircmd`, *or* `irmon`

When running `irouter`, `ircmd`, or `irmon`, you do not have to use double dashes when specifying command-line options:

```
irouter -n1 -shost2
```

For a list of the options that the JMQ router, command-line utilities, and JMQ applications can recognize, refer to **Chapter 3, "Command-Line Options,"** in this document.

This section describes the basic JMQ options that you need to understand to run `irouter`, `ircmd`, or `irmon`: the `-s`, `-n`, and `-w` options.

Specifying a Connection to Another Host: The -s Option

Use the `-s` option to specify the name of another router with which to form a connection.

If you start the router without an `-s` option, it forms a single standalone router. Applications can connect to this router, but other routers are not connected.

If you are starting a router, the `-s` flag tells the router to connect to a router on the indicated host. You must explicitly tell routers to connect to each other.

Specifying the Subnet to Use: The -n Option

The -n option specifies the subnet number to use for all communications between routers and applications. When using a subnet, you must start all JMQ applications and routers with this option.

Saving Current Options: The -w Option

Use the -w option to write the current options (except the -w option itself) to a defaults file. For more information on the defaults file, see the section, "JMQ Resource Files" in this chapter. Subsequent command-line invocations look for the default files, and use the options that are specified in them.

For example, if you started a router with the -w option as in the following example, the information would be stored in the defaults file:

```
irouter -sHost1 -n2 -w &
```

Because router startup information is stored in this defaults file, you do not need to provide router startup options again: subsequent router invocations start the router as if you had supplied the previous options.

For example, to subsequently invoke the router with these saved options, simply type:

```
irouter &
```

Specifying Command-Line Options to an Application

When you run a JMQ application within the JMQ environment, the general format for specifying command-line options is:

```
java program1 -- -shost1 -doprogram1.out
```

The java command runs the JMQ application, in this case program1. The double dashes separate the program options from the JMQ options that follow. The -s option specifies the name of the host to which to connect, and the -do option specifies an output file, here program1.out.

Configuring for Firewalls

If one JMQ application, application A, connects to JMQ application B, using subnet 1, packets from A to B have a dynamically allocated source port and a destination port of 9313.

Conversely, in the same scenario, packets from B to A have a source port of 9313 and the same dynamically allocated destination port.

You can use this rule to configure your firewall: you can allow traffic on the JMQ port through the firewall to a router inside the firewall. This is similar to configuring a firewall to enable ftp.

Router and JMQ Application Connection Behavior

Routers and JMQ applications behave in a specific manner within a JMQ network topology.

JMQ router behavior is governed by the following rules:

- Only one router can exist per host per subnet. See "Subnets", in this chapter.
- The router is not connected to another router when it starts up, unless you specify otherwise with the `-s` flag.
- The router uses subnet 0, unless you specify a different subnet with the `-n` flag.

JMQ applications are typically governed by the following rules and exhibit the following behavior; however, you do have some flexibility in your deployment in changing the behavior of your JMQ application. Because you have this flexibility, the rules that follow do not necessarily apply to all possible deployments.

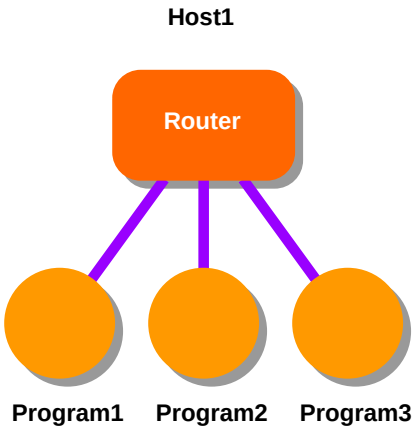
- The application attempts to connect to a router on the local host, unless you specify the `-s` flag. If you do, the application will attempt to connect to a router that is on a host specified.
- The application uses subnet 0, unless you specify the `-n` flag, which can include another subnet number (for example, `-n4`).
- The application has a connection to only one router at any time.

Running One JMQ Router on One Host

The first basic JMQ network topology is a router running on one host with JMQ applications connected to that router, as shown in FIGURE 1-1.

Note – JMQ applications do not connect directly together to communicate; rather, they communicate through one or more routers.

FIGURE 1-1 One Router on One Host with JMQ Applications Connected



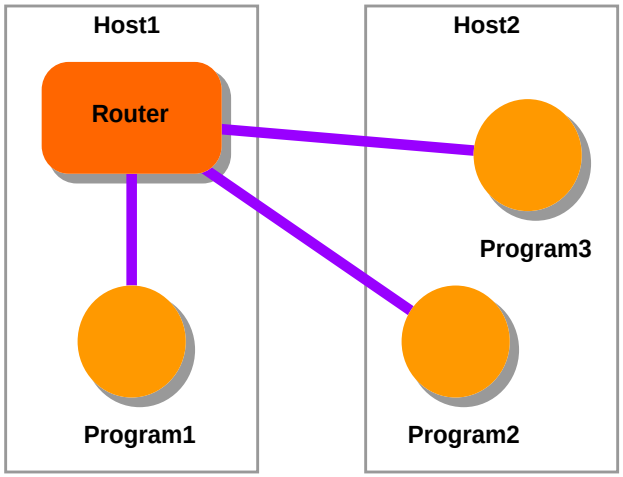
Use the following commands to start the router on one host and to start each JMQ application on the host:

```
irouter
java -classpath ... Program1
java -classpath ... Program2
java -classpath ... Program3
```

Running One JMQ Router on One Host with Local and Remote JMQ Applications

FIGURE 1-2 illustrates one router running on a host, with JMQ applications on that host connected to the router, and the router connected to JMQ applications running on the second host.

FIGURE 1-2 One Router and Two Hosts and JMQ Applications Running on Both Hosts



In FIGURE 1-2, one JMQ application, `Program1` resides on the same host as the router and is considered local to that router. Two other JMQ applications, `Program2` and `Program3`, reside on a second host remote to the router on the first host.

To run a router with both local and remote JMQ applications, use the following commands. The `-s` command specifies a connection between the JMQ applications running on `Host2` and the router running on `Host1`.

On `Host1`:

```
irouter
java -classpath ... Program1
```

On `Host2`:

```
java -classpath ... Program2 -- -sHost1
java -classpath ... Program3 -- -sHost1
```

Creating the JMQ Router Network: Multiple Routers in a Router Network

The previous examples show how to set up a JMQ system using simple JMQ network topologies. You can also create multiple routers in a larger and more complex router network.

Only fully-connected router topologies are supported.

Note – You should have an in-depth understanding of the various command-line options used during the startup of the router and of JMQ applications before you begin your router network design.

Designing and creating a complex JMQ system network involves three major steps:

- Design your network topology. Locate the hosts on which the JMQ applications and routers reside.
- Create your router network by:
  - Starting routers on the hosts specified in your routing network design, or by
  - Using `ircmd` to connect routers to match your design.

Once you have designed your JMQ network topology, you can start your JMQ applications that use the JMQ network as the means to transfer data.

Designing Host Placement and Connection Topologies

When creating topologies, you can create diagrams that show the connections or links through which message data travel. You should be aware of both the types of topologies and the relation between topology and performance.

When routers are connected together—for example, in a triangle configuration, message data can move from any router to any other router.

Specifying Hosts

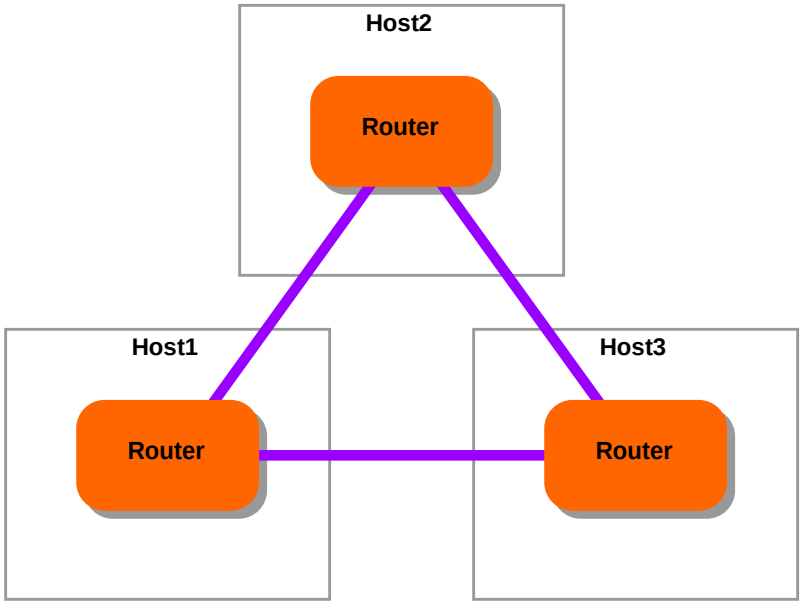
You can start and connect routers by:

- Starting a router without using a script and specifying on the command line all hosts with which it must communicate.
- Using the `ircmd` utility to connect two running routers.

The `-s` option specifies a comma-separated list of hosts to which the router connects. The list represents the connections upon which message data travels in your JMQ network.

For example, to build the fully-connected triangle network shown in FIGURE 1-3, you can start the routers with `irouter -s` specified.

FIGURE 1-3 Fully-connected Router Network



Running a Fully Connected Router Network

To run the fully-connected router network topology shown in FIGURE 1-3, use the following commands on the specified hosts.

On `Host1`:

```
irouter -sHost2,Host3
```

On the `Host2`:

```
irouter -sHost1,Host3
```

On `Host3`

```
irouter -sHost1,Host2
```

To connect two routers, you must ensure only that one of them connects to the other router at startup. You do not need to tell both routers to connect.

Note – If you need to start the same router network on a regular basis, you can use the `-w` command-line option, described in more detail in "The -w Command-line Option", to enable each router to remember its startup options between startups.

Examining Your Network Configuration

Once the routing network is running, you can use the `ircmd` utility to examine the configuration of the network and to detect any potential problems. The `ircmd` utility provides a variety of useful commands for monitoring and debugging JMQ applications. For more information on the `ircmd` utility, refer to Chapter 4.

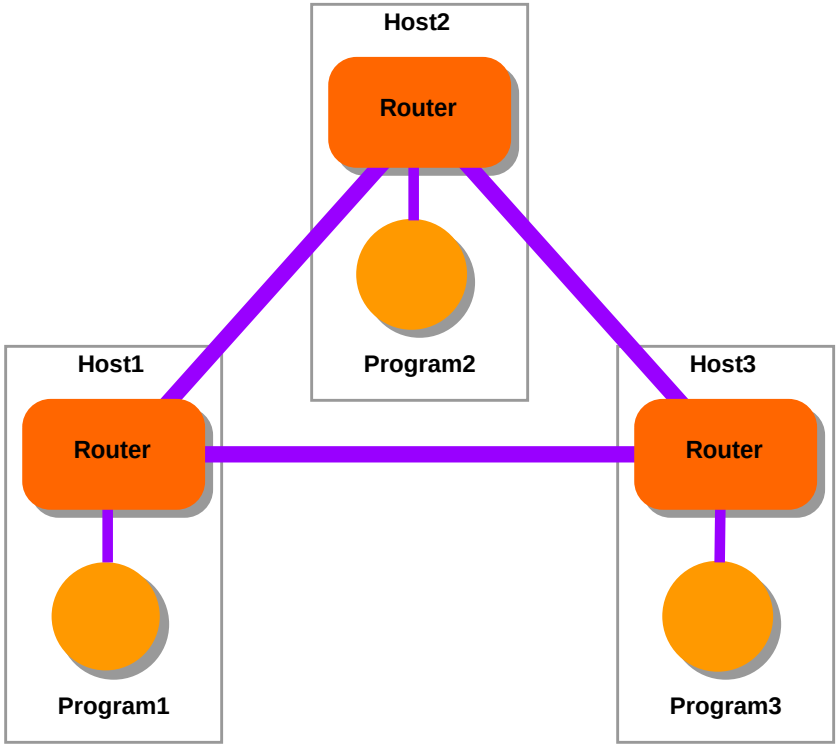
Starting JMQ Applications in a JMQ Network

Once your JMQ routing network is running, you can start JMQ applications on the hosts in your JMQ network. By default, JMQ applications attempt to connect to a router located on their local host (the host on which they reside).

Fully Connected Router Configuration With Applications

FIGURE 1-4 shows JMQ applications (programs 1 through 3) started in conjunction with a fully connected router configuration.

FIGURE 1-4 Fully-connected Router Configuration With JMQ Applications



In this example:

- Routers on `Host1` and `Host2` have JMQ applications (Program1, Program2, or Program3) on each host connected to them.
- The router on `Host3` is directly connected to the routers on the routers on the other two hosts; that is, it connects to the Router on `Host1` and `Host2`.

- The router connection enables each the connection of each JMQ application.

Running a JMQ Application in a Fully-Connected Router Topology

To run a JMQ application in a fully connected router topology, use the following example commands on the specified hosts:

On `Host1`:

```
irouter -sHost2,Host3
java -classpath ... Program1
```

On `Host2`

```
irouter -sHost1,Host3
java -classpath ... Program2
```

On `Host3`:

```
irouter -sHost1,Host2
java -classpath ... Program3
```

Creating Your Own JMQ Router Network

Once you understand JMQ router networks in general, you can start designing your own JMQ router networks.

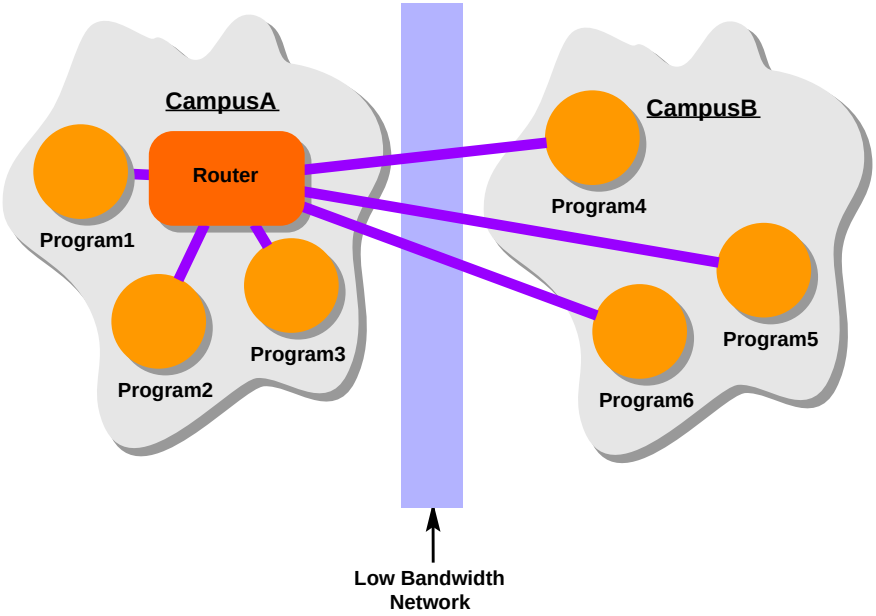
You typically adapt router topologies according to your business needs. The following two scenarios describe the various issues and criteria you might address when choosing a topology. These examples represent two advanced topologies but do not include the complete range of possible topologies.

One Router

Suppose a company has two campuses, CampusA and CampusB. This company has applications running on hosts on both campuses; these applications potentially need to send and receive messages between them.

In this scenario, there is one router located at the main campus, CampusA. Refer to FIGURE 1-5.

FIGURE 1-5 One-Router Campus Network



Advantages

The primary advantage to the one-router configuration is performance. Messages sent between programs entail a one-router hop. If an additional router existed in the configuration, messages would pass through the additional router, thus creating the potential for message delay. The presence of only one router guarantees that JMQ message performance is optimized.

Disadvantages

While the topology shown in FIGURE 1-5 represents a fundamentally sound design, there are two potential disadvantages to this design:

- Unnecessary traffic over a low-bandwidth network
- Lower fault tolerance for intra-campus messaging

Unnecessary Traffic Over a Low-Bandwidth Network

Potentially heavy message traffic might exist when traffic runs over a low-bandwidth network that connects two sites running the JMQ system. For example, when the router on CampusA must communicate with Program4, Program5, and Program6, traffic increases over the low- bandwidth network that connects the two campuses.

Lower Fault Tolerance for Intra-Campus Messaging

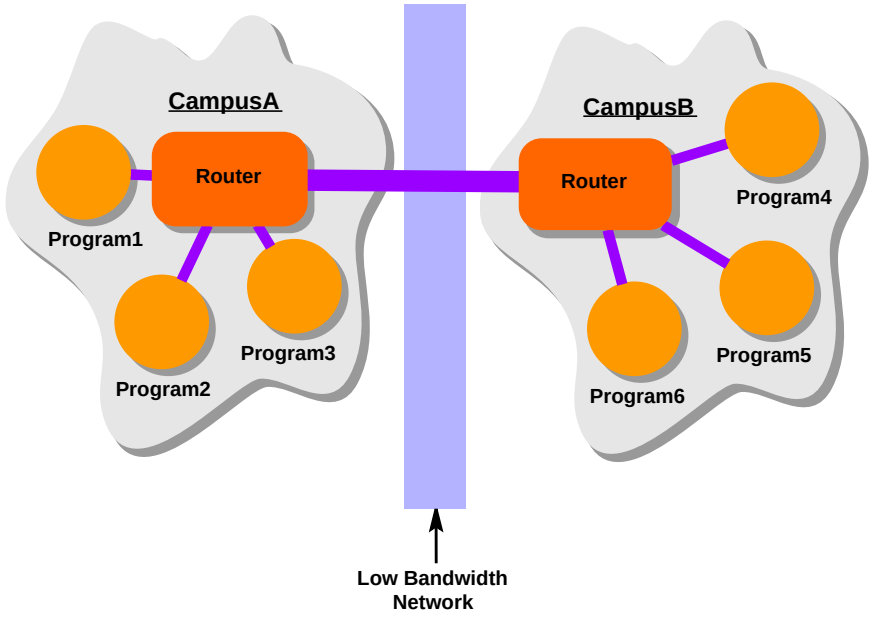
Messages might be lost if the router needs to be restarted. Creating a topology with a second router introduces a degree of fault tolerance, because the second router assumes the role of routing all messages to any interested JMQ application. However, in normal circumstances, JMQ performance might slow because of the second router.

Two Routers

The second scenario is a topology using two routers, one at CampusA and one at CampusB. Refer to FIGURE 1-6. The clouds on either campus in FIGURE 1-6, by convention represent the possible existence of multiple hosts; they indicate that applications can reside on one or many hosts within a campus.

In this example, communication between programs is at most a two-router hop. Progam1 for example, communicates with Program2 only through the router on CampusA.

FIGURE 1-6 Two-Router Campus Network



Advantages

A two (or multiple-) router topology has these possible advantages:

- Traffic reduction on a low-bandwidth network
- Router customization for each site
- Balancing of disk and CPU load

Traffic Reduction on a Low-bandwidth Network

The traffic on a low-bandwidth network that includes multiple routers is reduced relative to the traffic in a one-router topology. If `Program1` in FIGURE 1-6 sends a message that has a topic of interest to all other applications on either campus, the traffic across the low-bandwidth network is only between routers—that is, from CampusA router to CampusB router. Communication does not need to flow between all JMQ applications, as it does in the single-router configuration.

Router Customization Per Campus

The JMQ administrator can customize and configure the routers according to the needs of the campus (or site) where the router is located. This is substantially different from the one-router configuration shown in FIGURE 1-5, where the JMQ administrator configuring the router located on one campus and applications must attempt to meet the JMQ configuration needs of all campuses in the network topology.

Balancing of Disk and CPU Load

The multiple router approach enables the balancing of the disk and CPU load of the JMQ system across multiple routers and hosts. If a router at one campus fails, only traffic to and from that router is affected. For example, if the router on CampusA crashes, Program4, Program5, and Program6 can still communicate. The load is balanced across CPUs and disks because communication between JMQ applications is performed by an alternative router when one router fails.

With the one-router topology, all traffic between JMQ applications ceases if the one router in the JMQ network crashes.

In a multi-router configuration, the CPU load is also lessened because messages that are sent by and received by programs in the same campus do not need to leave the campus. For example, if CampusB had a topic that was of interest only to Program4, Program5, and Program6, a message sent by Program4 does not travel unnecessarily across the low-bandwidth network to the other campus.

Note – A star (one-router) topology might be optimal for the requirements of some businesses. You must base your JMQ topology upon your business and logistical requirements.

Monitoring a Large JMQ Router Network

The most convenient way to monitor a large router network is to use the router monitor application (`irmon`). This utility displays debugging messages from applications running on hosts connected to the same router network. You set message levels using the `-dn` command-line option.

If you run an application that has a GUI, you might need to write debug messages to a file, using the appropriate API, or you might specify that a JMQ administrator use `irmon` to examine the debug messages from your JMQ application.

For more information on monitoring a network, see Chapter 7.

JMQ Resource Files

System resource files are files that contain data related to the entire JMQ system. You often do not need to refer explicitly to system resources in the course of your normal use of JMQ; however, when running the router or JMQ applications, you should have a basic understanding of their use, location, and the order in which the JMQ router searches through the resource files.

System resource files include:

- System `.rfb` files. Contain the default and any changed settings for application and utility invocation.
- The system `res.ndx` file. Contains indices for all installed `.rfb` files.
- Per-router persistent messages. Created by the router running on a host and persist across multiple router invocations. These contain messages that are being sent between JMQ applications.

JMQ uses a simple, file-based persistent storage mechanism to store its default files, and data log.

The `-w` Command-line Option

If you use the `-w` option when starting the router, JMQ stores current command-line options (except the `-w` option itself) in a defaults file in the USER directory. (Refer to "Resource File Search" for a description of the USER directory.)

TABLE 1-1 lists router or utility commands and the default files in which options related to that command are stored.

TABLE 1-1 Commands and Default Files for Option Storage

Command	Default files
`irouter`	`rtrdef.rfb`
`ircmd`	`cmddef.rfb`
`irmon`	`cmddef.rfb`

If you use the `-w` option, you can return command option settings for each of these commands to their default values by first explicitly removing the `.rfb` file related to that command.

Alternatively, if you want to return the setting in the `.rfb` file to the default, you can specify the default command option setting when you run a command with the `-w` option.

Note – The `-w` option is cumulative: it adds to previous settings stored in the defaults file, and overwrites previous settings stored in the file, if those settings exist.

Resource File Search

The JMQ router locates resource files on a host by examining a specified resource file path. The location of resource files is platform-dependent.

The following information describes how JMQ searches for resources, although typically you do not need to be concerned with the search operation.

In a resource search operation. JMQ resource directories are designated either SYSTEM or USER. JMQ follows this pattern in searching for a resource file.

- The first directory in the list is the SYSTEM directory (or the least significant directory).
- The last directory item is the USER directory (or the most significant directory).

Information in the USER directory supersedes information in the SYSTEM directory (or augments the information in the former, depending on the JMQ operation that is performed.)

Where Resources Are Located on the Solaris Platform

SYSTEM resources are located under `/var/opt/SUNWjmq/res/sys`

USER resources are located under `/var/opt/SUNWjmq/res/user`

Both these directories must exist and the user-id of the running irouter process must be able to write to them. If either of these two directories are not accessible, JMQ uses the current directory (`"."`) for the SYSTEM directory in addition to the USER directory.

Problems in accessing SYSTEM or USER resources might indicate a bad or corrupt installation.

JMQ_HOME must be set. If JMQ_HOME is not set, the system uses the current directory (".") as the value of JMQ_HOME to find resource directories.

Where Resources Are Located on the Windows Platform

SYSTEM resources are located under `%JMQ_HOME%\res\sys`

USER resources are located under `%JMQ_HOME%\res\user`

Both these directories must exist and the running `irouter` process must be able to write to them. If either of these two directories is not accessible, the current directory (“.”) is used for SYSTEM as well as USER information for the duration of that session.

Problems in accessing SYSTEM or USER resources might indicate a bad or corrupt installation.

Overriding the Default Resources Location by Defining JMQ_RESPATH

You can override the default resources file location by using the JMQ_RESPATH environment variable to determine the location of resources on a particular system, although it is seldom necessary to override the location. You should set this environment variable only to change the default behavior of resource management—to specify different directories for SYSTEM or USER or both, and to specify additional paths to search for resources.

If you plan on providing paths to additional directories to search for resources, they must exist and be writable by the JMQ router.

If you define JMQ_RESPATH, it must contain at least two file paths, separated by specific platform-dependent path separators. The first path is assumed to be SYSTEM and the path one is assumed to be USER. You can include additional directories in JMQ_RESPATH.

If the directories specified in JMQ_RESPATH are not accessible, the directory “.” is used; that is, the directory that is the current directory relative to the JMQ executable that is running.

If you need to add an additional path to search in the Solaris operating environment, set JMQ_RESPATH as in the following:

JMQ_RESPATH to `/var/opt/SUNWjmq/res/sys:/new/place/to/look:/var/opt/SUNWjmq/res/user`

If you need to add an additional path to search on Windows, set:

JMQ_RESPATH to `%JMQ_HOME%\res\sys;\new\place\to\look;%JMQ_HOME%\res\user`

For more information on setting these variables, see the “Environment Variables” section in this chapter.

Persistent Router Message Store

The JMQ router creates a directory apWWXXYYZZ.<routersubnet> in the SYSTEM or sys directory as a location to store persistent messages (messages that persist between subsequent router invocations).

WWXXYYZZ is the host IP address n hexadecimal notation on which the router is running. Thus, if the router is running on host IP address 129.144.252.101, JMQ converts the address

```
129.144.252.101 --> 81.90.fc.65 --> 8190fc65 --> ap8190fc65.1
```

<routersubnet> is the subnet of the running router. Thus, ap8190fc65.1 is the persistent message store for a router on subnet 1.

Cleaning Up Remaining Persistent Router Messages

To clean up leftover persistent messages, delete the apWWXXYYZZ.<routersubnet> directory when the router is shutdown.

If you are experiencing problems with a JMQ application or JMQ router, and before contacting Sun Support, stop the router, remove the appropriate apWWXXYYZZ.<routersubnet> directory and run the router again.

You can facilitate the removal of the apWWXXYYZZ.<routersubnet>directory when debugging your applications by:

- Setting JMQ_RESPATH before you run the router (for example, setenv JMQ_RESPATH ‘pwd‘, which sets the resource file path that contains the persistent message directory to the directory specified in ‘pwd‘).
- Removing the apWWXXYYZZ.<routersubnet> directory between each run of the router.

Changing Persistent Message Store

You can modify JMQ_RESPATH to change persistent message store to remedy a lack of disk space at the default location or to take advantage of a faster (less busy) disk as a location for JMQ_RESPATH.

To store persistent messages in another location, change the SYSTEM respath by setting JMQ_RESPATH, following the guidelines in “Overriding the Default Resources Location by Defining JMQ_RESPATH”.

You must set both the SYSTEM (the sys respath) and the USER component (the user respath) when you change the persistent message store:
JMQ_RESPATH=/fastdisk/jmq/res/sys:/var/opt/SUNWjmq/res/user

This setting establishes persistent message store at the /fastdisk SYSTEM location and the /var/opt USER location.

Solaris Umask Settings and Alternate Users

Resources created by JMQ on Solaris machines in the Solaris operating environment use the user’s umask settings. All common resources (res.ndx) and directories are created with umask mode settings of 666 and 777 respectively.

In a typical deployment, the same user (that is, root or another standard user created to run JMQ routers) runs the router each time it is invoked.

However, the default umask settings might cause problems if another user tries to run the router, or install a JMQ resource. Execution will fail if the appropriate permissions do not exist.

To avoid potential problems, you can use `chmod` to change the mode on `res.ndx` files, and other files created in `/var/opt/SUNWjmq/res/sys` and `/var/opt/SUNWjmq/res/user`.

Obtaining Router Option Information

If you want to see the options that are either passed in the command line or that the JMQ system reads in from the `rtrdef.rfb` files that are found on `JMQ_RESPATH` and which are the router uses, you can invoke the router with the `-ds1` option. Refer to the following examples.

Starting the Router Using Default Values

Invoking the router with the `-ds1 option` displays the following response, and displays the initial default option values as `Cmd Flags`:

```
host1% ./irouter -ds1
Initializing router...

Java(tm) Message Queue              Copyright 1999-2000
Version: 1.1 ( Build R-186 )        Sun Microsystems, Inc.
Compile: Fri Apr 14 04:34:40        All Rights Reserved
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
New process 0x33ee...
Cmd Flags: -c0 -ds1 -frtrdef.rfb
...
```

Starting the Router, Reading from User Resource rtrdef.rfb File

The router reads from the `USER` resource rfb file if you do not specify other options.

If `/var/opt/SUNWjmq/res/user/rtrdef.rfb` (the user resource rfb) has values `-n6 -ds1`, invoking the router by entering `irouter` without options provides the following response, which displays the values in the user `rfb` file, the initial `-c` (licensing) value, and the name of the `rtrdef.rfb` file:

```
host1% ./irouter
...
Cmd Flags: -n6 -c0 -ds1 -frtrdef.rfb
...
```

JMQ Connection Enablement

The JMQ product specifies the number of connections which can be enabled for use with the router at any given time. The allowable number of concurrent connections to a router is determined by the product versions you install. In Java Message Queue 1.1, for example, the maximum enabled connections are as follows:

- Developer Edition allows a maximum of 5 connections per router.
- Business Edition and Business Edition 10-pack allows an unlimited number of connections per router.

Unlimited means that the JMQ product itself does not set an upper limit on the number of concurrent connections. However, in reality, a limit is imposed by the underlying operating system.

Note – The number of router instances you are allowed to run is determined by the number of router licenses included in the product version you obtain.

Running one router on one subnet and another router on a second subnet requires you to have two licenses, irrespective of the placement of the subnets on the same host or different hosts.

JMQ controls connection enablement through special files. These files are installed as part of the JMQ product and have the following filename convention:

```
JMQ_HOME/lib/lic/irouter*.lic
```

where `JMQ_HOME` is the location of the installed JMQ product and * is an identifying sequence number that JMQ assigns.

Connection enabler files are not cumulative. Only one connection enabler file can be used by a given router at any given time. For example, if you installed one edition of the JMQ product that had an enabled connection limit of 5 then upgraded to another edition of the product that had an enabled connection limit of 500, you could run the router with either a 5 connection limit or a 500 connection limit, but you could not get a 505 connection limit.

Listing or Specifying the Number of Enabled Connections

To query the number of enabled connections (available connection-enabler files), pass only the `-C` option to the router. For example:

```
irouter -C
```

In addition to the enabled connections information, the -C option might provide other information (such as the date a trial license and its enabler file will expire).

To specify the number of enabled connections when starting a router (that is, to specify which enabler file to use), pass the -C option plus the name of the connection enabler file. For example:

```
irouter -C irouterunlim.lic
```

Note – If more than one enabler file exists when a router is started but a specific enabler file is not specified, the enabler file allowing the greatest number of connections is used.

For more information on options that can be passed when starting the router, refer to **Chapter 2, "Command-Line Options,"** in this document.

Error Conditions and Messages

JMQ reports the following types of error conditions related to connections and licensing when you invoke a router:

- Refused connections, maximum number of connections has been reached.
- Invalid connection-enabler (`.lic`) files.
- Problems finding or opening connection-enabler files.
- If you have not set `JMQ_HOME`, and JMQ is searching in another location for a license file.
- Invalid expiration date.
- Trial expiration has expired.
- Trial expiration will expire.

JMQ Runtime Considerations

You should be aware of the following considerations when running your JMQ system. For more information on facts about JMQ to keep in mind when designing or running your JMQ system, see the JMQ FAQ at `http://www.sun.com/workshop/jmq/faq.html`.

Message Storage in Multi-router Topologies

If you have a multi-router topology:

- Persistent messages are stored on the router connected to the sending client.

- Messages held for delivery to a durable subscriber are stored on the router connected to the receiving client.

Client Connecting to Same Router for Durable Messages

If you have a multi-router topology and you:

- Create a durable subscriber that is connected to Router A.
- Make the subscriber inactive so that messages need to be held for delivery when the subscriber reconnects.

Your client must connect back to the *same* router to obtain those messages once the client is active again (that is, your client must connect to Router A, and not to Router B).

Timeout on Persistent Messages

When a persistent JMQ message has a time to live (TTL) value associated with it, the clock pertaining to the time to live will restart after a router restart

For example, if your JMQ client application sends a message with a 30-second TTL, and after 23 seconds the JMQ router with which the JMQ client application is communicating is restarted, the messages will persist for another 30 seconds.

Because this behavior only affects persistence, the client application code will not receive an expired message—that is, the JMQ client application will receive the message and then discard it.

Timeout on Acknowledgment from Router

There is a 30 second timeout, by default, on any JMQ client when it is connected to a router to receive an acknowledgement from the router that a message has been successfully sent.

Because a router might be loaded with message activity to the extent that it cannot provide this acknowledgment within this timeout value, you might increase this timeout value via the `-t` option on the application command-line or when you configure a `ConnectionFactory` object with `jmqconfig`. See the description of the `acktimeout` attribute in TABLE 4-2 in **Chapter 4, "Configuring JMQ Administered Objects,"** for more information.

Fragmentation Size for Messages

For the purposes of efficiency and reliability, JMQ fragments large messages into 7K sections on both the client and the router.

Chapter 2:

Command-Line Options

The Java Message Queue software provides a special set of options that you can specify with the JMQ `irouter`, `ircmd` and `irmon` and with your own JMQ applications. Use these options to configure various communications, debugging, and output features. You can specify these options at the command line or use parameters to pass them to an application.

Options for the `irouter`, `ircmd`, *and* `irmon`

TABLE 2-1 lists options that you can specify with `irouter`, `ircmd`, and `irmon`.

TABLE 2-1 Command-Line Options for `irouter`, `ircmd`, and `irmon`

Argument	Description
`-C`	Lists the available connection licenses. For example: • `5` - a 5 connection router • `unl` - an unlimited connection router This command-line option applies only to the `irouter` utility.
`-C`*connLicType*	Sets the number of connections that can be enabled. Entering `-C` alone at the command prompt provides the possible values for *connLicType*. To set the connections, provide one of the connection licenses to the `-C` option. For example: `% -C25` This command-line option applies only to the `irouter` executable.
`-do`*outdev*	Sends output to the given device. The argument, *outdev*, must be `stdout`, stderr, `null`, or a file name.

TABLE 2-1 Command-Line Options for irouter, ircmd, and irmon

Argument	Description
-dn#	Specifies the printing level output to the network. You must provide a print level of 0 or greater. This option is usually used in conjunction with irmon to monitor applications remotely. This option is ignored by applications that are not connected to remote processes. See Chapter 9 for information on print groups and printing levels. JMQ specifically uses the network level of a group to determine if a message is sent to then etwork monitor, irmon; the message is only sent if the network level is greater than or equal to debug level of the message. Unlike the time level, the network level is independent of the print level.
-ds#	Specifies the printing level for system-defined groups. You must provide a print level of 0 or greater. See Chapter 9 for information on print groups and printing levels.
-dt#	Sets the timer output level. Use the timer output option in conjunction with the print output options to display a timestamp value on each print line for each operation that has occurred. You must provide an output level of 0 or greater. The timer and network levels for both system and user groups are initialized withthe same default values. The time level of a group is used to determine if a time stamp is appended to the front of a message. Both the print level and the timer level must be appropriate for a message to be printed. If the print level indicates the message should not be printed, you should not specify a time level (that is, a timestamp) for that message. For example, a -du specification that is at level 3 will not be printed. However, the timer level must match the debug level and the print level must be less than the debug level. Thus, both the timer level and the print level must be greater than or equal to the message debug level. See Chapter 9 for information on print groups and printing levels.

TABLE 2-1 Command-Line Options for irouter, ircmd, and irmon

Argument	Description
-du#	Sets the printing level for user-defined print groups only. You must provide a print level of 0 or greater. See Chapter 9 for information on print groups and printing levels.
-e*outdev*	Sends error message output to the given device. All error messages are sent to this device. The argument, *outdev*, must be null, stdout, or a file name.
-f*filename*	Specifies a startup file from which to load the defaults settings. The default file is read from each of the directories that are set in the JMQ_RESPATH environment variable. For example, if JMQ_RESPATH = /tmp:/home/jdoe, JMQ looks for the filename in /tmp/filename ./filename /home/jdoe/filename If JMQ_RESPATH is not set, JMQ searches for filename in the default locations in this order: /var/opt/SUNWjmq/res/sys/filename ./filename /var/opt/SUNWjmq/res/user/filename Do not prefix the name of the directory to this option; for example, if you entered irouter -f/home/jdoe/mydefaults, JMQ appends the argument to the directories described above. You should rely on the mechanism shown above, instead.
-h	Prints information about this set of command-line options.
-n#	Sets the subnet number. See Chapter 1 for a discussion of valid subnet numbers. The subnet is used by the system to determine port numbers to use to connect processes together. If this option is not specified, the system defaults to subnet 0. All JMQ applications connected together must use the same subnet.

TABLE 2-1 Command-Line Options for irouter, ircmd, and irmon

Argument	Description
-q	Sets quiet mode. No startup banner is displayed.
-s*host1,host2...*	Specifies the hosts to connect to. You should separate host names with commas. Do not place spaces between host names. This option tells the utility on which machine to find a router to which it can connect. If this option is not specified, ircmd or irmon looks for a router on their own host by default. irouter does not connect to any host if -s is not used.
-w	the default settings to a sttartup file. All of the current command-line options (except -w) are stored in a default settings file.

Specifying Options for irouter, ircmd *or* irmon

To specify an option for a JMQ utility or the router, give the desired arguments after the command name. For example:, the following options specify a print level and a subnet to use.

```
%irouter -ds1 -n5
```

Specifying a File to Which to Write Console Output: The -do *Option*

Following starting the irouter, use the -do option to write console output to the file name immediately following the -do flag.

The following example runs the irouter, and directs console output to irouter.out.

```
irouter -doirouter.out
```

Specifying Error Output: The -e Option

The -e option provides a mechanism for directing the output of JMQ error messages. Valid output destinations are provided in the following table.

TABLE 2-2 Error Output Options and Destinations

-e option Value	Result
stdout	Sends error output to stdout. This is the default when -e is not specified.
null	Does not display error output.
filename	Save error output in the specified file (for example, -elog.out).

You can specify log files to capture JMQ errors by placing the file name of the error log after the command that starts the irouter. The log is placed in the directory from which you run the command.

To specify an error log for the irouter:

```
irouter -eerrlog
```

Obtaining Information About Connections

If you need to observe information about JMQ system connections, you can specify debugging level one (using -ds1) when you run the router and JMQ applications.

Specifying this debugging level causes the router and the applications to print a message when a connection is made or broken. For example, the following command starts a router with debug level one set on subnet 5. The command to start the JMQ application Program4 behaves in a similar fashion.

```
irouter -ds1 -n5 -sHost2 -eerrlog.Host3 &
java -classpath ... Program4 -- -ds1 -n5
```

If debug level one is set, messages from the router when a connection closes are:

```
Channel closed...
Process 0x6e3f on host granite went away...
```

The JMQ router provides the following message when it opens a connection, in this case to Host1:

```
Connect succeeded with host host1
```

When the router terminates, it displays the following message:

```
Router shutting down.
```

You can write debug output to a file, `outfile`, with the `-do` option. In this example, the router is started and debug level one is indicated. The router runs on subnet 5. It sends debug output to `outfile` and the error log to an error log on `Host3`.

```
irouter -ds1 -dooutfile -n5 -sHost2 -eerrlog.host3
```

Options for JMQ Applications

TABLE 2-3 lists options that you can pass to JMQ Java applications.

TABLE 2-3 Command-Line Options for JMQ Applications

Argument	Description
`-do`*outdev*	Sends output to the given device. The argument, *outdev*, must be a file name.
`-ds#`	Specifies the printing level for all print groups, including both system-defined and user-defined groups. You must provide a print level of 0 or greater. For more information on using print groups, see Chapter 9.
`-n#`	Sets the subnet number. See Chapter 1 for a discussion of valid subnet numbers. The subnet is used by the system to determine optional port numbers (or the equivalent) to use to connect processes together. If this option is not specified, the system defaults to subnet 0. All JMQ applications connected together must use the same subnet.

TABLE 2-3 Command-Line Options for JMQ Applications

Argument	Description
`-shost`	Specifies the host to which to connect. This option tells the application on which machine to find a router to which it can connect. By default this option is not specified, the JMQ application looks for a router on its own host. Note that you can specify only one host for the application to find a router to which to connect; you can specify more than one host after the `-s` option when using the router.
`-t`	Specifies the acknowledgement timeout value in milliseconds.Default is 30000 (30 seconds). 0 indicates no timeout; that is, the JMQ client application will wait indefinitely for acknowledgements from the router. For more information on the acknowledgement timeout value, see TABLE 4-2 in Chapter 4.
`--`	Separates JMQ application options from JMQ system options.

Specifying Options for a JMQ Application

For a JMQ application, you can specify options that are passed to the underlying system. To specify an option for a JMQ application, you must precede the desired arguments with a double dash (`--`). For example:

```
%java app -- -ds1 -n5
```

You can pass the string array of the command-line options to the JMQ `QueueConnectionFactory` or `TopicConnectionFactory` method that takes a string array, a `java.util.Properties` instance, or you can use applet parameters by passing the applet instance.

For more information on the APIs of the `ConnectionFactory` classes, see the Javadoc supplied with the JMQ distribution.

For applets, you can pass the command-line options through parameters. When using JMQ, use the following parameters for applets:

```
param name = -n value = "subnetnumber"
param name = -s value = "hostname"
param name = -do value = "filename"
```

The following example sets up a parameter list to connect to the host, `myhost.mydomain.com` on subnet 3:

```
param name = -n value = "3"
param name = -s value = "myhost.mydomain.com"
```

Chapter 3:

JMQ Router and Supporting Utilities

The Java Message Queue software provides router and other supporting utilities that you can use to run and debug Java Message Queue applications:

- `irouter`. The JMQ Router. Routes messages delivered by applications and utilities to another router or client.
- `irmon`. The JMQ Monitor. Displays debugging and router network information.
- `ircmd`. A command interface that monitors behavior and displays status and administrative information on the router network.

The command-line utilities named above use the JMQ options listed in Chapter 2. The syntax for these utilities is provided in this chapter.

The Java Message Queue provides additional utilities that assist you in configuring your JMQ application environment and in running JMQ. These utilities are discussed in subsequent chapters, and include:

- `jmqconfig`. The JMQ Config utility that you can use to configure and manage JMQ Administered Objects.
- `jmqadmin`. The JMQ Admin utility that you can use to administer JMQ Queues and JMQ client application authentication at a JMQ router.
- `jmqsvcadmin`. The JMQ Windows Service Admin utility (for Windows NT and Windows 2000 Services only) that is used to install or remove the JMQ Router Service.

`irouter`

Syntax

`irouter` [*command-line options*]

Description

Starts the router, which forwards messaging information between connected hosts and applications. The behavior of the router can be controlled and monitored using the `ircmd` and the `irmon` command-line utilities.

Several of the JMQ command-line options are useful for monitoring and debugging routers. The debugging, error logging, and network output commands can be used to redirect router messages to log files or redirect the messages over the network to a monitoring station.

Command Options

See Chapter 2 for a complete list of JMQ command-line options that you can specify with `irouter`.

See Also

For information on `ircmd`, see the `ircmd` section later in this chapter.

---

irmon

Syntax

`irmon` [*command-line options*]

Description

Starts the monitor application, which displays debugging and network output. You can use any of the standard JMQ debugging options to redirect the output to a file. You can display router information in the following ways:

- Specify the `-dn`, `-ds`, or `-du` command-line options with `irmon` to display the output in the monitor. Use these options to view router information when the router is run remotely.
- Specify one of the `set debug` *hostname* subcommands with `ircmd` to display router information at the command-line prompt.
- Specify the `-dn`, `-ds`, or `-du` command-line options with `irouter` to display router information at the command-line when the router is run locally.

Note that the monitor application does not display all the information displayed by the system print groups. For example, you would see more information when running `irouter -ds4` than you would if you were running `irouter -dn4` in conjunction with the monitor application. The monitor application separates the output from each process by a banner that indicates its `PHYSICAL` JMQ address. Refer to Chapter 9 for more information on system debugging output.

Command Options

See Chapter 2 for a complete list of JMQ command-line options that you can specify with `irmon`.

ircmd

Syntax

`ircmd [-r` *scriptfile*`] [`*command-line options*`]`

Description

Runs `ircmd`, which enables you to perform queries and administrative tasks on the router network. Use this router command-line utility to:

- Query each router about its current connections and interests
- Reconfigure the connections between routers
- Verify connections to routers
- Control network print output
- Display various status information

Before messaging information can be transmitted to or from a router, you must connect to the router. To connect to a router, include the JMQ command-line option `-n` to specify the subnet. The subnet must be the same as the subnet on which the router is running. Use the `open` subcommand to open a connection to the router running on the same subnet. When the connection is open, you can query the router and issue commands, such as reconfiguration and control commands.

Command Option

The `ircmd` command takes the option `-r` *script-file*, which loads the sequence of commands from the given script and runs them. If a script file is not given, a command prompt appears enabling you to enter commands.

Note – You must include a space between the `-r` option and the *scriptfile* argument.

See `Chapter 2, "Command-Line Options,"` for a complete list of JMQ command-line options that you can specify with `ircmd`.

Subcommands

The ircmd command takes the following subcommands:

addpath *directory*

Appends the specified directory to the current setting of JMQ_RESPATH. Changes to JMQ_RESPATH become effective in the current instance of ircmd.

connect *host1* to *host2*

Creates a new connection from the router on host1 to a router on host2. The host2 must be connected directly or indirectly to the ircmd program. You can determine whether the host is connected to ircmd by running the hosts subcommand and verifying that its name is in the hosts list. The unconnected host must already have a router running on it, and the router must be on the same subnet that you are currently using.

connections *hostname*

Shows the connections to the router on the specified host. Each connection is given by the name of the host on the other end of the connection and the process ID of that host. The process ID consists of two numbers: [*ip-address:pid*] for TCP/IP where:

- *ip-address* is the IP address of the host.
- *pid* is the process ID.

The possible connection types are:

- LOCAL. Router runs on the same machine as an application and the application is internal to the router.
- REMOTE. Router runs on a different machine than the application or the application is external to the router.
- PRIVATE. Router-to-application connections.
- PUBLIC. Router-to-router connections.

The STATE value indicates whether the connection is:

- NEW. The connection has been initialized but no connection maintenance messages have passed through it.
- ALIVE. The router has actively pinged that connection.
- DEAD. The connection has been terminated.

The TimeDelta value indicates the time differential (in milliseconds) measured between the clocks on the sender and the recipient hosts. The QSize indicates the number of pending objects in the queue to be sent through that connection.

For example:

```
cmd>connections granite
1) (application) on granite [0xccfdd115,0x70ed]
     Type:PRIVATE,REMOTE State:ALIVE TimeDelta:0 QSize:0
     Num Sessions: 1
```

If you are using Windows NT, all applications communicate through a single JMQ dynamic link library rather than separately. For example, if you have three applications on a Windows NT machine, the router shows only one connection to that machine.

`help` [*subcommand*]

Lists all the valid subcommands and their syntax. If a subcommand is provided, displays information for that subcommand only.

`hosts`

Shows which hosts have routers that are connected to the current router. Only hosts that are running a router process are listed.

`interests` *hostname*

Gets the list of interests from the router on the specified host. The interests list is divided into local and remote interests. A `LOCAL` interest is an interest of a routine in the routing process itself. A `REMOTE` interest is the interest of another router. A remote interest is displayed as follows:

```
type: REMOTE hostid = hostid inthandle = handle, publicity
```

The *hostid* is given as [*ip:pid*] for TCP/IP:

- *ip* is the IP address of the host
- *pid* is the process id of the process
- *handle* is the interest handle
- *publicity* is either `PUBLIC` (an interest of a routing process) or `PRIVATE` (an interest of a client application).

The properties of a `LOCAL` or a `REMOTE` interest are displayed on the `template`:*proplist* line where *proplist* is a list of items of the form:

propid = {*prop-val*}

where

- *propid* is the property ID of the property

- *prop-val* is the value for the property

For example, the property values for PT_DESTDOMAIN and PT_DATACLASS are JMQ addresses.

```
cmd>interests granite
Interests for host granite: PHYSICAL:granite.sun.com
___
    type: LOCAL
    flags:
    template:
     PT_DESTDOMAIN := { PHYSICAL:granite.sun.com%DEF_CLASS:132 }
     PT_DATACLASS := { IARtrInterest }
     PT_EXPRID := { [0x2, 0x1] }
-------------------------------------------
    type: REMOTE hostid = [0xccfdd115,0x5b3] inthandle = -2, PRIVATE
    flags: ACKED, DURABLE
    qsize: 1
    name: MyDurInt
    template:
     PT_DATACLASS := { UInt32 }
```

`open` *hostname*

Opens a socket connection to the router running on the given host on the same subnet as `ircmd`. You must *always* open a connection to a router on a host before you can query or send commands to the router. See the Note below.

Note – When using the `-s` option with the `ircmd` utility, you must first connect to a host using the open *hostname* command.

`ping` *hostname*

Verifies the connection to the router on another host and provides the length of time it takes for a message to be sent from `host1` to `host2` and back. Time is shown in milliseconds. The most critical time values are the sent time, the received time, and the trip time.

`query info` *hostname*

Retrieves information about the application on the given host and displays it.

```
cmd>query info granite
Instance information for: PHYSICAL:granite.sun.com
    Class: (none) Instance: (none)
    Address: PHYSICAL:granite.sun.com/DEF_CLASS:28294
    Exec Path: irouter
```

A PHYSICAL address can be specified for the host name. The command is sent to all applications matching that address.

`quit`

Exits the `ircmd` program.

`script` *scriptfile*

Runs the given script file. The script file is an ASCII file that contains valid `ircmd` commands. For example, you could create a script file `pinghost.scr` that contains the command `ping granite`. When you execute the script, `pinghost.scr`, you obtain the following output:

```
cmd>script pinghost.scr
Ping time to granite: 0.5 sec
```

`set`

Enables you to remotely reconfigure processes and routers on a network. Currently, you can set only the print levels.

`set debug`

Configures the print group levels and output of processes on the network. A PHYSICAL address can be specified for the given host for all `set debug` commands. The command is sent to all applications matching that address.

`set debug` *hostname* `netlevel #`

Sets the network print level of the user and system groups of the router running on the given host.

`set debug` *hostname* `syslevel #`

Sets the network print level of the user and system groups of the router running on the specified host.

`set debug` *hostname* `usrlevel #`

Sets the user print groups level of the router running on the specified host. For example, the following command sets the user print level for all addresses matching PHYSICAL:granite.sun.com/exclass:* to 3:

```
cmd>set debug PHYSICAL:granite.sun.com/exclass:*
    usrlevel 3
Setting debug level of
    PHYSICAL:granite.sun.com/exclass:* to 3
Set successful
```

`setpath` *directory*

Sets the environment variable, JMQ_RESPATH, to the specified directory. The change is valid only for the current `ircmd` session.

`show respath`

Displays the active resource path setting.

`wait` *waittime*

Sets the maximum time (in seconds) to wait for a response object. If you receive a `No response!` message to an `ircmd` command, increase the wait time. The default wait time is 2 seconds.

See Also

For information on `irouter` and `irmon`, see the `irouter` and `irmon` sections in this chapter.

Chapter 4: Configuring JMQ Administered Objects

JMQ Administered Objects

JMQ Administered Objects enable developers to be independent from JMS service providers. With the `jmqconfig` utility, you can configure JMQ-specific Administered Objects that you store in a JNDI service provider. See the "Types of JMQ Administered Objects"discussion below.

The developer can then use the object names specified with `jmqconfig`, look up the objects and retrieve them and use them in JMQ applications.

Before you run the `jmqconfig` utility, you need to ensure that both you and the developer writing the code agree on the JMQ Administered Object name, placement, and hierarchy.

The Java Directory Naming and Directory Interface™ (JNDI) provides names and contexts for these JMQ administered objects. For more information on JNDI and JMQ, refer to the *Java™ Message Queue QuickStart Guide*. For more information on JNDI, see the JNDI Tutorial at `http://java.sun.com/products.`

See section 4.2 of the JMS Message Specification, Version 1.0.2, included with this distribution, for more information on administered objects.

Note that the JMQ product distribution does not include an LDAP server. You must obtain your own LDAP server if you choose to use an LDAP service provider.

JMQ supplies Sun Microsystem's LDAP service provider in addition to Sun Microsystem's Filesystem service provider (currently in Beta).

The jmqconfig Utility

Use the `jmqconfig` utility to perform the following operations on `Destination` or `ConnectionFactory` objects:

- Store a new JMQ administered object in a service provider
- Delete a JMQ administered object from a service provider
- Show the configuration of JMQ administered objects in a service provider
- List JMQ administered objects in a service provider

Usage Notes

Before you use the `jmqconfig` utility, you need to be aware of the following:

- You should quote all arguments, unless the argument consists of just one word.
- When you use the `-d` command to delete a JMQ Administered Object, the utility asks for confirmation before performing the operation. It then lists the contents of the object you are deleting.
- You can run the jmqconfig utility in non-interactive or scripting mode, using the `-f` option on any command. This is useful if you are running a script to configure JMQ Administered Objects and you do not want to be prompted when the script performs an operation.
- The individual running the `jmqconfig` utility must have appropriate permission for accessing the Directory Service; the permission needed depends on the operation the user wants:
  - reading—for listing and showing administered objects
  - writing—for storing administered objects
  - deleting—for deleting administered objects.

Types and Attributes of JMQ Administered Objects

JMQ administered objects you configure through the `jmqconfig` utility have attributes—informtion associated with the object. An attribute consists of an attribute identifier (name) and one or more attribute values. An attribute of a JMQ administered object can be mandatory or optional; if mandatory, you must specify the attribute when creating the object.

Types of JMQ Administered Objects

There are two types of JMQ Administered Objects for which you can supply attributes. Use the `jmqconfig` utility to configure the following types of objects:

- Destinations:
  - `Queues`
  - `Topics`
- ConnectionFactories
  - `QueueConnectionFactory`
  - `TopicConnectionFactory`

Attributes of JMQ Administered Objects

The following tables provide information on the attributes you can configure for each of the types of JMQ Administered Objects.

Notes on the Following Tables

Note the following about these tables:

- Attribute names are specified with the `-o` option, described under "jmqconfig Utility Syntax", below.
- One attribute for the `Destination` object is mandatory (that is, you must specify the name of a Topic or a Queue for a Destination object), but all attributes for the `ConnectionFactory` are optional.
- In TABLE 4-2, each attribute name has a default value that `jmqconfig` uses if you do not specify an attribute value.
- TABLE 4-2 also provides the allowable values and default values for the `JMSX` properties. Note that if you set a specific `JMSX` property to `True` you are enabling it in a message.

  For example, if you specify `JMSXUserID=true`, and thus store that attribute in the administered `ConnectionFactory` object, the `JMSXUserID` property is set in all messages sent. In this example, the value associated with the `JMSXUserID` property is the identity of the user sending the message.

TABLE 4-1 `Destination` Object Attributes

Attribute name	Description	Allowable values	Default value
name	Queue or Topic	The name of a Queue or a Topic. For example, `-o "name=myTestTopic"`	You must specify the name of a Topic or Queue for a Destination object.

TABLE 4-2 `ConnectionFactory` Object Attribute Names

Attribute name	Description	Allowable values	Default value
ackTimeout	A non-negative integer (in milliseconds) that specifies the amount of time the JMQ client application will wait before it times out for the acknowledgement from the router when a reliable persistent message is sent. When you configure this acknowledgement timeout value on an administered `ConnectionFactory` object, all connections that are generated from this factory object use this value.	Range is any positive integer values greater than 0 for a definite wait period. A 0 value specifies an indefinite wait period. You might want to increase the default value through your JMQ client application, if: • There is a router with which your JMQ client application communicates that might become busy, or • There is a slow connection that the JMQ client application might encounter when sending messages. For example, `-o “ackTimeout=40000”`	30000 milliseconds (30 seconds).
host	The host name for this router.	For example, `-o “host=myhost”`	localhost

TABLE 4-2 `ConnectionFactory` Object Attribute Names

Attribute name	Description	Allowable values	Default value
subnet	The port number of the router. The port that Java Message Queue uses is (9312 + n), where n is the Java Message Queue subnet number. For example, subnet = 10 translates to port number = 9322. Valid subnet numbers are 0 to 999. Java Message Queue also allows the user to specify the port number directly by passing a negative number for the subnet number. For example, subnet = -10 translates to port number = 10.	For example, `-o "subnet=5"`	0 (that is, port 9312)
`securityPort`	The port that is used by the JMQ client application for client authentication with the Client Authentication Server.	The port number must be the absolute port number and must be a positive integer. The port number must be associated with a free port on the server that runs the JMQ router and the `jmqadmin` utility. If you set the port to a value between 0 and 1024, the `uid` of the `jmqadmin` process must be root. For example, `-o "securityPort=21300"`	22000

TABLE 4-2 ConnectionFactory Object Attribute Names

Attribute name	Description	Allowable values	Default value
Any of the following JMSX properties: JMSXUserID JMSXAppID JMSXProducerTXID JMSXConsumerID JMSRcvTimestamp	For information on JMSX properties, see the JMS 1.0.2 specification.	true or false, t or f Note that you control the presence of a JMSX property in a message by setting this value to True or False. For example, -o "JMSXUserID=true"	False. JMSX* properties are not set in messages.

jmqconfig *Utility Syntax*

The following section shows the syntax for performing the tasks related to using the jmqconfig utility.

Storing a JMQ Administered Object to a Directory Service

Following is the syntax of the jmqconfig utility for storing a JMS Administered Destination object. Note that there is only one attribute, name, for this object:

```
jmqconfig
   -a [-f] -t <dest_type> -n <JNDI_lookup_name> -o name=<dest_name>
   -i <JNDI_initial_context_factory_name> -u <JNDI_provider_url>
   [-s <authentication_scheme>]
   [-p <principal (user name)>]
   [-c <credentials (password)>]
```

Following is the syntax of the jmqconfig utility for storing a JMS Administered ConnectionFactory object. Attribute names for ConnectionFactory objects are provided in Table 4-2, "ConnectionFactory Object Attribute Names," above:

```
jmqconfig
   -a [-f] -t <conn factory type> -n <JNDI_lookup_name>
   -i <JNDI_initial_context_factory_name> -u <JNDI_provider_url>
   [-o attribute_name=<attribute_value>] [-s <authentication_scheme>]
   [-p <principal (user name)>]
   [-c <credentials (password)>]
```

Refer to the following table for an explanation of the syntax for these commands and options.

TABLE 4-3 Options for Storing a JMQ Administered Object to a Directory Service

Option	Description
-a	Add a JMQ administered object.

-t <*type*>	The type of the JMQ administered object.
	type can be any one of:
	`t=Topic`
	`q=Queue`
	`qf=QueueConnectionFactory`
	`tf=TopicConnectionFactory`

-n <*JNDI_lookup_name*>	The name the application program will use to lookup the JMQ administered object from the Directory Service. The name must be unique within the JNDI context specified.
-o "<*attribute_name*> <*attribute_value*>"	The name, value pairs that follow each `-o` command line option configure individual attributes in the JMQ Administered Objects Refer to Table 4-1, "Destination Object Attributes," or Table 4-2, "ConnectionFactory Object Attribute Names," for attribute specifications.
-f	Specifies non-interactive or scripting mode. By specifying this flag, you run the `jmqconfig` utility in non-interactive mode; if the authentication information is missing or incorrect, `jmqconfig` returns an error message and exits. This behavior is useful when you do not want to be prompted when running a script to configure administered objects.
-i <*JNDI_initial_context_factory_name*>	Specifies the provider-specific initial context factory for JNDI.
-u <*JNDI_provider_url*>	Specifies the provider-specific URL for JNDI.

TABLE 4-3 Options for Storing a JMQ Administered Object to a Directory Service

Option	Description
-s <*authentication_scheme*>	Sets the `java.naming.security.authentication` property. This value specifies the authentication scheme used by the service provider. The utility supports `none` and `simple`.
-p <*principal*>	Sets the `java.naming.security.principal` property. This value specifies the principal (user name) used to authenticate the user at the service provider end.
-c <*credentials*>	Sets the `java.naming.credentials` property. This value specifies the credentials (password) used to authenticate the user at the service provider end.

Examples

To store a `Topic` object where the JNDI lookup name is `myTopic` and the topic name is 'This is my Test Topic', using LDAP server as the JNDI Service Provider:

```
jmqconfig -a -t t -n “cn=myTopic” -o “name=This is my Test Topic"
  -i “com.sun.jndi.ldap.LdapCtxFactory” -u “ldap://mydomain.com:389/o=JMQ”
```

To store a `Topic` object where the JNDI lookup name is `myTopic` and the topic name is `This is my Test Topic`, using a Filesystem Directory Service:

```
jmqconfig -a -t t -n myTopic -o “name=This is my Test Topic”
  -i “com.sun.jndi.fscontext.RefFSContextFactory” -u “file:/tmp”
```

To store a `QueueConnectionFactory` where the JNDI lookup name is `myQCF`, the host is `foo`, and the subnet is 5, using an LDAP server as a Directory Service:

```
jmqconfig -a -t qf -n “cn=myQCF” -o “host=foo” -o “subnet=5”
  -i “com.sun.jndi.ldap.LdapCtxFactory” -u “ldap://mydomain.com:389/o=JMQ”
```

To store a `QueueConnectionFactory` where the JNDI lookup name is `myQCF`, the host is `my_machine`, the subnet is 5, using an LDAP server as a Directory Service, the `simple` security scheme is used with a principal (username) of `foo`, credentials (password) of `foo`, and an acknowledgment timeout value of 40 seconds or 40,000 milliseconds is specified:

```
jmqconfig -a -t qf -n “cn=myQCF” -o “host=my_machine” -o “subnet=5”
    -i “com.sun.jndi.ldap.LdapCtxFactory” -u “ldap://mydomain.com:389/o=JMQ”
    -s simple -p "uid=foo, ou=JMQObj, o=JMQ" -c foo -o “ackTimeout=40000”
```

Deleting a JMQ Administered Object from a Directory Service

To delete a JMQ administered object, use the following syntax:

```
jmqconfig
     -d [-f] -n <JNDI_lookup_name>
     -i <JNDI_initial_context_factory_name>
     -u <JNDI_provider_url> [-s <authentication scheme>]
     [-p <principal (user name)>] [-c <credentials (password)>]
```

TABLE 4-4 Options for Deleting a JMQ Administered Object from a Directory Server

Option	Description
-d	Delete a JMQ administered object.
-n <*JNDI_lookup_name*>	The name a JMQ application program uses to look up the JMQ administered object from the Directory Service. The name must be unique within the JNDI context specified.
-f	Specifies non-interactive or scripting mode. By specifying this flag, you run the `jmqconfig` utility in non-interactive mode; if the authentication information is missing or incorrect, `jmqconfig` returns an error message and exits. The `jmqconfig` utility queries the content of the deleted JMS administered object when you use `-f`; if you realize after running the command that it was a mistake to delete a particular object, you can subsequently recreate it.
-i <*JNDI_initial_context_factory_name*>	Specifies a provider-specific initial context factory for JNDI.
-u <*JNDI_provider_url*>	Specifies the provider-specific URL for JNDI.
-s <*authentication_scheme*>	Sets the `java.naming.security.authentication` property. This value specifies the authentication scheme used by the service provider. The utility supports `none` and `simple`.
-p <*principal*>	Sets the `java.naming.security.principal` property. This value specifies the principal (user name) used to authenticate the user at the service provider end.

TABLE 4-4 Options for Deleting a JMQ Administered Object from a Directory Server

Option	Description
-c <*credentials*>	Sets the `java.naming.credentials` property. This value specifies the credentials (`Password`) used to authenticate the user at the service provider end.

Example

To delete a JMQ administered object without a confirmation step where the JNDI name is `myTopic` and the LDAP server does not have an authentication scheme established:

```
jmqconfig -d -f -n “cn=myTopic” -i “com.sun.jndi.ldap.LdapCtxFactory”
  -u “ldap://mydomain.com:389/o=JMQ”
```

Showing the Configuration of a JMQ Administered Object in a Directory Service

To display the configuration of a JMQ administered object, use the following syntax:

```
jmqconfig
  -q [-f] -n <JNDI_lookup_name> -i <JNDI_initial_context_factory_name>
  -u <JNDI_provider_url> [-s <authentication scheme>]
  [-p <principal (user name)>] [-c <credentials (password)>
```

TABLE 4-5 Options for Showing the JMQ Administered Objects in a Directory Service

Option	Description
-q	Display JMS administered objects.
-f	Specifies non-interactive or scripting mode. By specifying this flag, you run the jmqconfig utility in non-interactive mode; if the authentication information is missing or incorrect, `jmqconfig` returns an error message and exits.
-n <*JNDI_lookup_name*>	The name the JMQ application program uses to look up the JMQ administered object from the Directory Service. The name must be unique within the JNDI context specified.
-i <*JNDI_initial_context_factory_name*>	Specifies a provider-specific initial context factory for JNDI.

TABLE 4-5 Options for Showing the JMQ Administered Objects in a Directory Service

Option	Description
-u <*JNDI_provider_url*>	Specifies the provider-specific URL for JNDI.
-s <*authentication_scheme*>	Sets the `java.naming.security.authentication` property. This value specifies the authentication scheme used by the service provider. The utility supports `none` and `simple`.
-p <*principal*>	Sets the `java.naming.security.principal` property. This value specifies the principal (user name) used to authenticate the user at the service provider end.
-c <*credentials*>	Sets `java.naming.credentials` property. This value specifies the credentials (password) used to authenticate the user at the service provider end.

Example

To show a JNDI name where the binding name is `myTopic`:

```
jmqconfig -q -n "cn=myTopic" -i "com.sun.jndi.ldap.LdapCtxFactory"
  -u "ldap://mydomain.com:389/o=JMQ"
```

Listing the JMQ Administered Objects in a Directory Service

Use the following syntax to list objects in a directory. You can specify the type of objects you want to list:

```
jmqconfig -l
  [-f] [-t <type>] -i <JNDI_initial_context_factory_name>
  -u <JNDI_provider_url> [-s <authentication scheme>]
  [-p <principal (user name)>] [-c <credentials (password)>
```

TABLE 4-6 Options for Listing the JMQ Administered Objects in a Directory Service

Option	Description
-l	List a JMS administered object.

TABLE 4-6 Options for Listing the JMQ Administered Objects in a Directory Service

Option	Description
-f	Specifies scripting or non-interactive mode. By specifying this flag, you run the `jmqconfig` utility in non-interactive mode. If the authentication information is missing or incorrect, `jmqconfig` returns an error message and exits.

-t <*type*>	Type of the JMS administered object where type can be (`q,t,qf,tf`):
	`q=Queue`
	`t=topic`
	`qf=QueueConnectionFactory`
	`tf=TopicConnectionFactory`
	If you do not provide the `-t` flag, `jmqconfig` lists all JMQ administered objects in the context.

-i <*JNDI_initial_context_factory_name*>	Specifies a provider-specific initial context factory for JNDI.
-u <*JNDI_provider_url*>	Specifies the provider-specific URL for JNDI.
-s <*authentication_scheme*>	Sets the `java.naming.security.authentication` property. This value specifies the authentication scheme (mechanism) used by the service provider. The utility supports `none` and `simple`.
-p <*principal*>	Sets the `java.naming.security.principal` property. This value specifies the principal (user name) used to authenticate the user at the service provider end.
-c <*credentials*>	Sets the `java.naming.credentials` property. This value specifies the credentials (password) used to authenticate the user at the service provider end.

Example

To list objects in an LDAP server:

```
jmqconfig -l -i “com.sun.jndi.ldap.LdapCtxFactory”
  -u “ldap://”mydomain.com:389/o=JMQ”
```

Displaying a Help Page About the `jmqconfig` *Utility*

To display a help page about the `jmqconfig` utility, enter:

```
jmqconfig -h
```

`jndi.properties` *File*

When you run `jmqconfig`, you need to supply the `-i` and the `-u` options to ensure that the provider-specific initial context factory and URL are specified.

However, you can optionally edit the `jndi.properties` file to provide these values.

The `jndi.properties` file is subject to the placement restrictions noted below:

- The `jndi.properties` file is located in the `/var/opt/SUNWjmq/admin/` (Solaris platform) or `%JMQ_HOME%\admin\` (win32).
- The `jmqconfig` command includes the path to `jndi.properties` in its CLASSPATH, so when using JMQ with JDK 1.2, you do not need to make any changes to your CLASSPATH.
- However, when you use JDK1.1.x, the file must be copied to `$JAVA_HOME/lib/` on the Solaris platform or to `%JAVA_HOME%\lib \` (win32) directory, to ensure that it is read by the JNDI.

Service Provider Authentication Considerations

Service providers have authentication schemes set up so only users with appropriate permissions can perform operations on objects stored in them.

When you use the `jmqconfig` utility to configure administered objects, you can use either one of two authentication schemes:

- `none`. There is no authentication scheme for the service provider. One example of this scheme is used when you use a Filesystem service provider.
- `simple`. The user must first provide the user name and password to authenticate himself. If both match those of the service provider, the user is granted the permission to perform the action.

To use this scheme, you must first set up the service provider so it also uses `simple` as its underlying authentication scheme.

The service provider you use can have different permission rules, depending on the different types of operations per user.You can specify authentication schemes, Principal (username) and Credentials (password) at the command-line, which is recommended, or have them available in the `jndi.properties` file.

Refer to the syntax tables in this chapter for more information on setting permissions by configuring security on an administered object.

Service Provider Authentication Defaults

`jmqconfig` applies service provider authentication defaults in the following manner:

- If you have not specified any authentication-related properties in the `jndi.properties` file or at the command-line, `jmqconfig` defaults to the `none` authentication scheme.
- If you specify the `username` but do not specify the authentication scheme, the scheme is set to `simple`, if `simple` is supported by your service provider.

Because the Principal(`username`) and Credentials (`password`) values are ignored when the authentication scheme is explicitly set to `none`, you should set the authentication scheme to `none` if you want to ensure that other authentication properties that might have been set are ignored.

If you supply an empty string for the `password`, the authentication scheme is set to `none`, regardless of the setting of the authentication property.

If you specify the authentication scheme in the `jndi.properties` file and set it to `simple`, but did not specify the Principal (`username`) and password values, the user is always prompted for Principal (`username`) and Credentials (`password`). If the user performs an action that does not require any authentication (for example, if you have set up a service provider so *any* user can query an object) the user can ignore any prompting for Principal(`username`) and Credentials (`password`) by pressing the Return key twice.

If you specify the authentication scheme and the `username` in the `jndi.properties` file and the authentication scheme is set to `simple`, the user is prompted for `password` only. If the action the user wants to perform does not require any authentication, the user can ignore any prompting by pressing the Return key.

If the authentication scheme, Principal (`username`), and Credentials (`password`) are specified in the `jndi.properties` file and the authentication scheme is set to `simple`, the user does *not* receive a prompt for Principal (`username`) and Credentials (`password`).

jmqconfig Error Messages

The folllowing section lists `jmqconfig` error messages, and provides a problem description and a solution for each error message.

Note that some errors, when they occur, include a response to "Please type jmqconfig -h for help."

Insufficient Arguments Specified

Problem. You typed jmqconfig without arguments.

```
Error: Insufficient arguments
```

Solution. Supply the correct number of arguments to `jmqconfig`. Type `jmqconfig -h` for help in ascertaining the correct arguments.

Problems Initializing Initial Context

Problem. `jmqconfig` could not initialize the initial context because you either did not correctly specify the `jndi.properties` file, or you did not specify the `-i` or `-u` options at the command line.

```
Error: Problems initializing the initial context. Please make sure to
use -i and -u options to set java.naming.factory.initial and
java.naming.provider.url properties at the command-line.
```

Solution. Ensure that you either specify the `-i` and `-u` options set to the correct java.naming.factory.iniial and java.naming.provider.url specifications, or that you supply a correct jndi.properties file.

Missing Value for Name Argument

Problem. You did not supply a value for the destination name argument.

```
Error: Missing destination name (-o name) argument.
```

Solution. Supply a value for the destination name argument (-o name=value).

Password Not Specified In Script Mode

Problem. You ran `jmqconfig` in script mode, but did not supply a password. `jmqconfig` responds that you do not have permission to perform the action. `jmqconfig`'s response depends on the authentication required to do an action you specify.

```
Error: You don't have permission to perform this action in this context.
```

Solution. Ensure you specify the authentication password at the command-line for any action for which authentication is required, when running `jmqconfig` in script mode.

Invalid Authentication Scheme Specified

Problem. You specified an invalid authentication scheme.

```
Error: The specified authentication scheme is not supported.
```

Solution. Ensure you provide a valid authentication scheme appropriate for your service provider.

Attempting to Access an Object Not a JMQ 1.1 Administered Object

Problem: You have attempted to access a JMQ Administered Object that was either created with the Early Access Release of JMQ 1.1 or was not a JMQ Administered Object.

```
Error: Error this is a JMQ administered object created using jmqconfig
from the Early Access release of JMQ 1.1 or it is not a JMQ administered
object. If it is a JMQ administered object from the Early Access release
of JMQ1.1, please remove it as it is not compatible with this release of
JMQ.
```

Solution. If the object was created using `jmqconfig` from the Early Access release of JMQ 1.1, remove the object. Otherwise, ensure that you access only JMQ Administered Objects.

Chapter 5: JMQ Administration Console

The Java Message Queue product provides the `jmqadmin` utility to assist in the administration of the Java Message Queue system. The Java Message Queue system consists of messages, applications, and routers. The graphical interface provided by the `jmqadmin` utility is called the Administration Console. This chapter discusses the jmqadmin utility and Administration Console from the following perspectives:

- Purpose explains why you would use the Administration Console.
- Special Considerations notes things you should know before using the Console.
- Administration Tasks provides a listing of the most common administrative tasks you can perform with the Console.

The `jmqadmin` utility is but one of several utilities provided with the Java Message Queue product. The other utilities are: `jmqconfig` (for configuring administered objects); `jmsvcadmin` (for installing the router as an Windows Service—Windows NT and 2000 only); `ircmd` and `irmon` (for controlling, debugging, and monitoring the JMQ router when running Java Message Queue applications).

For information on `jmqconfig`, see the "Configuring JMQ Administered Objects" chapter. For information on `jmqsvcadmin`, see the "The Router as a Windows Service" chapter. For information on `irouter`, `irmon`, or `ircmd`, see the "JMQ Router and Supporting Utilities" chapter.

Purpose

The Administration Console provides a graphical interface for the following administrative tasks:

- Deleting Queues and their persistent messages.
- Enabling the authentication of users to the router.
- Managing the logs that record administrative events.

Special Considerations

Understanding the following characteristics and behaviors of the Administration Console will enable you to make the best use of the tool:

- Running the Administration Console (`jmqadmin`) as a different user may prevent it from working properly.

The Administration Console periodically saves out information into files. One example of this is the list of users and passwords from the Administer Users panel. In the Solaris operating environment, where it is possible for different users to have different file creation and access permissions, it is recommended that the Administration Console (`jmqadmin`) be run as a single user only.

For example, if the Administration Console is run initially as the user `admin`, it should always be run as the user `admin`. Running the Administration Console as a different user may prevent it from working properly. The files that are saved out by the Administration Console (`jmqadmin`) as user `admin` may not be readable or writeable by a different run of Administration Console as a different user.

- The Administration Console can delete only Queues (and their persistent messages). Durable Topics cannot be deleted using the Administration Console.

  Further, the Administration Console can list only Queues for deletion that have (or had) receivers. This does not mean that you cannot delete Queues that do not have receiver, but that you must know the name of such a Queue in order to delete it. The Administer Queues panel provides a textfield for entering the name of a Queue to delete if it is not shown in the list of Queues.

  Because deleting a Queue that does not have a receiver requires that you know the name of the Queue, it is very important to keep track of all Queues created. By doing so, Queue deletion will not pose a problem.

  For information on deleting a Queue and its persistent messages, see "Deleting a Queue" on page 73.

- The Administration Console does not poll Queue status—you must use the Refresh button to obtain updated views.

- The logs created by the Administration Console contain administration events only.

  Every time the Log Manager in the Administration Console is started, a new log file is created. Entries to the log reflect administration messages and events (such as the starting of the client authentication server). Log entries do not include messaging system events (such as stopping the router).

  For additional information on Administration Console log files, see "Enabling Client Authentication" on page 75.

- The Client Authentication Server is for development only.

  The Java Message Queue product includes a simple, non-encrypted client authentication mechanism. When it is enabled, user names and passwords must be passed in with any Java Message Queue applications requesting to connect to the router. However, the decision to use client authentication is made by the *client application developer*, not the Administrator. (The developer chooses whether to use a JMS API that requires a username and password.) Further, the usernames and passwords entered into the Administration Console are stored in *clear text*.

For these reasons, the Client Authentication Server should not be relied upon in a deployment situation that has real security requirements.

For information on setting up client authentication through the Administration Console, see “Enabling Client Authentication” on page 75. For information on writing client applications that use the simple authentication mechanism, see the Quickstart Guide.

Administration Tasks

As mentioned previously, the `jmqadmin` utility provides a graphical user interface called the Administration Console. The Administration Console provides support for common administrative concerns such as: cleaning up persistent messages for Queues, enabling authentication of users accessing the router, and managing administration event logs. This section is divided into the following task categories:

- “Using the Console” on page 71 gives instructions for starting, setting up, and stopping the Administration Console.
- “Managing Queues and Persistent Messages” on page 73 provides details on deleting Queues and their associated persistent messages.
- “Enabling Client Authentication” on page 75 describes the nature and contents of the logs, as well as how to start, stop, and time logging.
- “Enabling Client Authentication” on page 75 discusses how to manage usernames and passwords for authenticating client applications requesting to connect to the router.

Using the Console

The Administration Console is an easy-to-use utility through which you can perform a number of administrative tasks. Certain tasks, like deleting Queues, would be quite tedious without the aid of the Administration utility.

Note – The Administration Console performs on a per router basis rather than system-wide basis. The Administration Console must be run on the same host and subnet as the router being administered. Also, any files created or stored as a result of the Administration Console functions will be relative to the given machine on which the router resides.

Launching the Administration Console

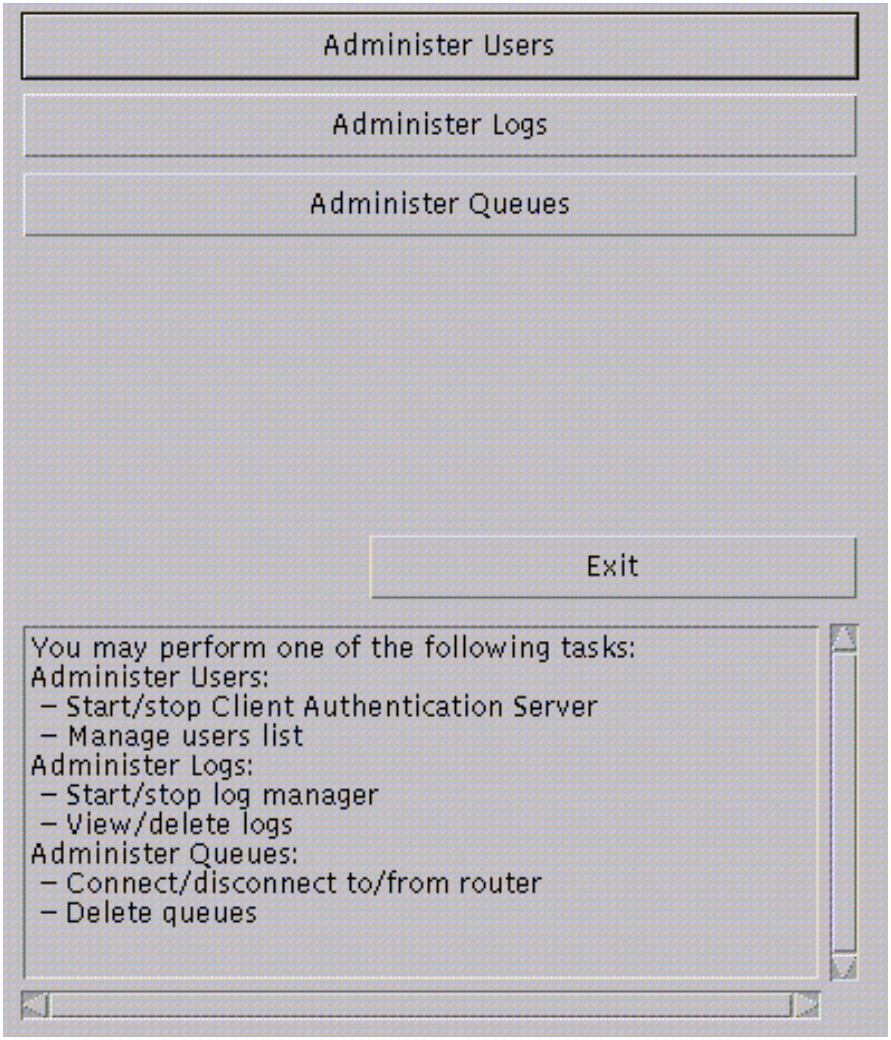
To launch the Administration Console from a command prompt, simply type:

```
jmqadmin
```

The main panel of the Administration Console will be displayed (FIGURE 5-1).

Alternatively, you can launch the `jmqadmin` utility from the directory viewer for your system by double-clicking on the file icon. (On a Windows NT system, for example, you can launch the Console by double-clicking the `jmqadmin` file icon in either a Windows Explorer or My Computer window.)

FIGURE 5-1 Main Administration Console Panel



Closing the Administration Console

To close the Administration Console (thereby terminating the running `jmqadmin` process), simply click Exit on the main panel. Closing the window in which the Administration Console is running also terminates the process.

Managing Queues and Persistent Messages

In the Administration Console, queues and their associated persistent messages are treated as one entity. Topics cannot be managed through the Administration Console at this time. This section provides instructions for performing tasks related to Queues and their persistent (stored) messages.

Creating a Queue

In general, there is no need to explicitly create a Queue using the Administration Console and so such functionality is not present. If a message is sent (or a receiver set) for a Queue that does not already exist, the named Queue is automatically created.

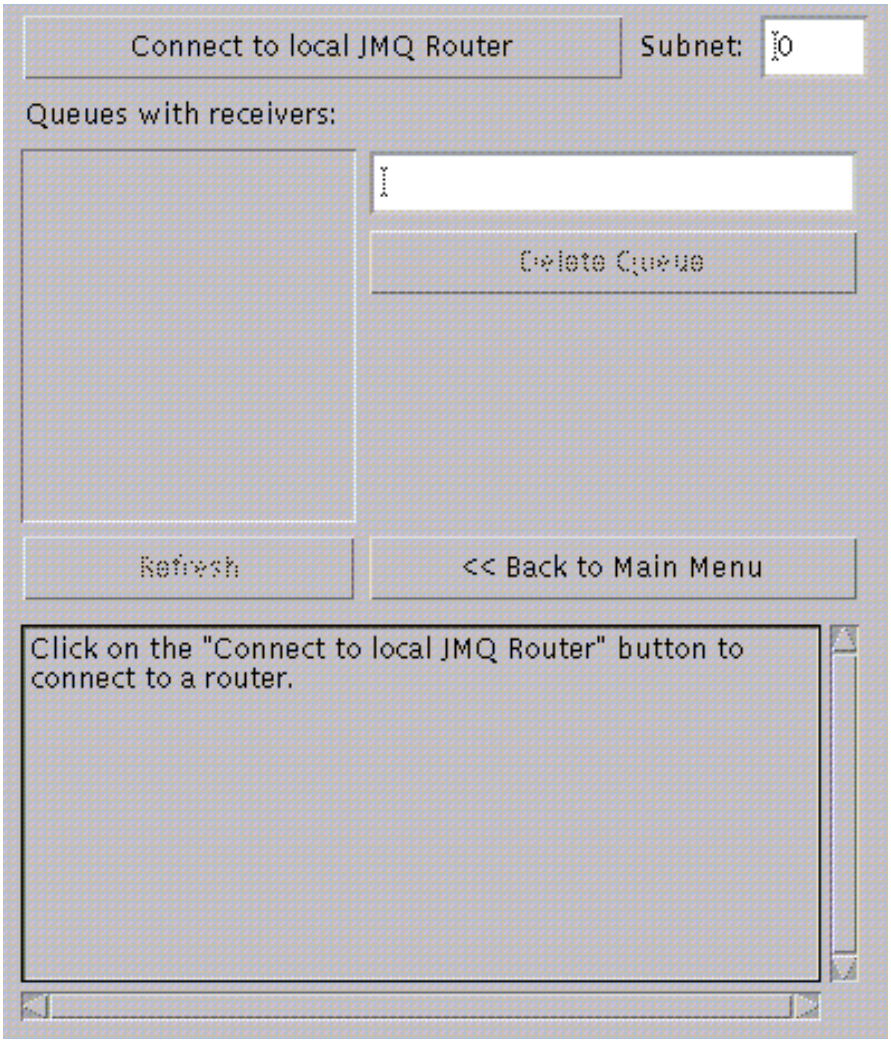
For consistency and ease of administration, it is a good idea for the Administrator of the Java Message Queue system to provide a naming convention for Queues and keep a list of Queue once created. In some cases (such as the deletion of Queues which have no receivers), knowing the name of an existing Queue is important.

Note – Queues can also be created programmatically by client application. However, this approach is not recommended because it impacts the portability of the client application between JMS providers. For more information on creating provider-independant applications, see the QuickStart's Creating Client Applications chapter.

Deleting a Queue

1. If the Administration Console is not already running, launch it.
2. Click Administer Queues. A panel similar to FIGURE 5-2 will be displayed.
3. Click Refresh to update the Queues with Receivers list.
4. Indicate the subnet on which the router to be administerd is running, then click Connect to JMQ Router.
5. Indicate the Queue to be deleted:
   - If the Queue to be deleted appears in the Queues with Receivers list, select it.
   - If the Queue to be deleted *does not* appear in the Queues with Receivers list but its name is known, enter that name in the textfield provided.
6. Click Delete.

FIGURE 5-2 Administer Queues Panel



Persistent messages are stored in the `res` subdirectories of the Java Message Queue installation. Examination of the filename or contents of a persistent message file will not indicate the name of the queue to which a message file belongs. That means, if the Queue does not appear in the queue list on the Administer Queues panel, there is no simple way to delete it or its persistent messages without running the risk of deleting persistent messages for other queues.

Note – Do not remove the `res` directory or its `sys` and `user` subdirectories from the Java Message Queue installation. The router stores persistent messages and files needed for its own operation (such as saved startup options) in those directories. If the directories do not exist, the router will store its files in the directory from which the command to start it is issued (current working directory).

Deleting a Queue's Stored Persistent Messages

To delete a Queue's persistent messages, simply delete the Queue itself—the persistent message store is automatically erased

Enabling Client Authentication

The Administration Console enables the authentication of client applications by providing a graphical interface to the Client Authentication Server. When authentication is enabled, user names and passwords must be passed in with any Java Message Queue applications requesting to connect to the router. If the user name or password are not recognized, the application does not connect to the router. Otherwise, a connection to the router is established.

Using the Administration Console, you can start or stop the Java Message Queue Client Authentication Server. You can also add the user names and passwords that will be passed in with Java Message Queue applications requesting to connect to the router. One default user is provided with a username of `Administrator` and a password of `password.`

The client authentication mechanism provided with the Java Message Queue product is for development only. In this implementation, developers of client applications (not Administrators) determine whether authentication will be used by selecting the appropriate JMS API. Also, even when client application developer chooses authentication, usernames and passwords are stored by the Administration Console as clear (unencrypted) text in an easily identifiable file. That file is located as follows:

```
/var/opt/SUNWjmq/admin/Users.data (Solaris)

$JMQ_HOME/admin/Users.data (Windows)
```

where `$JMQ_HOME` refers to the environment variable containing the location of the installed Java Message Queue product.

Note – Java Message Queue 1.1 cannot read the Users.class file written out by Java Message Queue 1.0 which contained usernames / passwords.

Briefly, to enable client authentication for the Java Message Queue messaging system, perform the following steps:

1. Create an client application that uses the JMS APIs requiring username/password authentication. (For details on creating an application of this nature, see the QuickStart "Creating Client Applications" section.)
2. Start a router.
3. Start the Client Authentication Server.
4. In the Administration Console, add a username and password for each user who will request access to the router.
5. Run the client application.

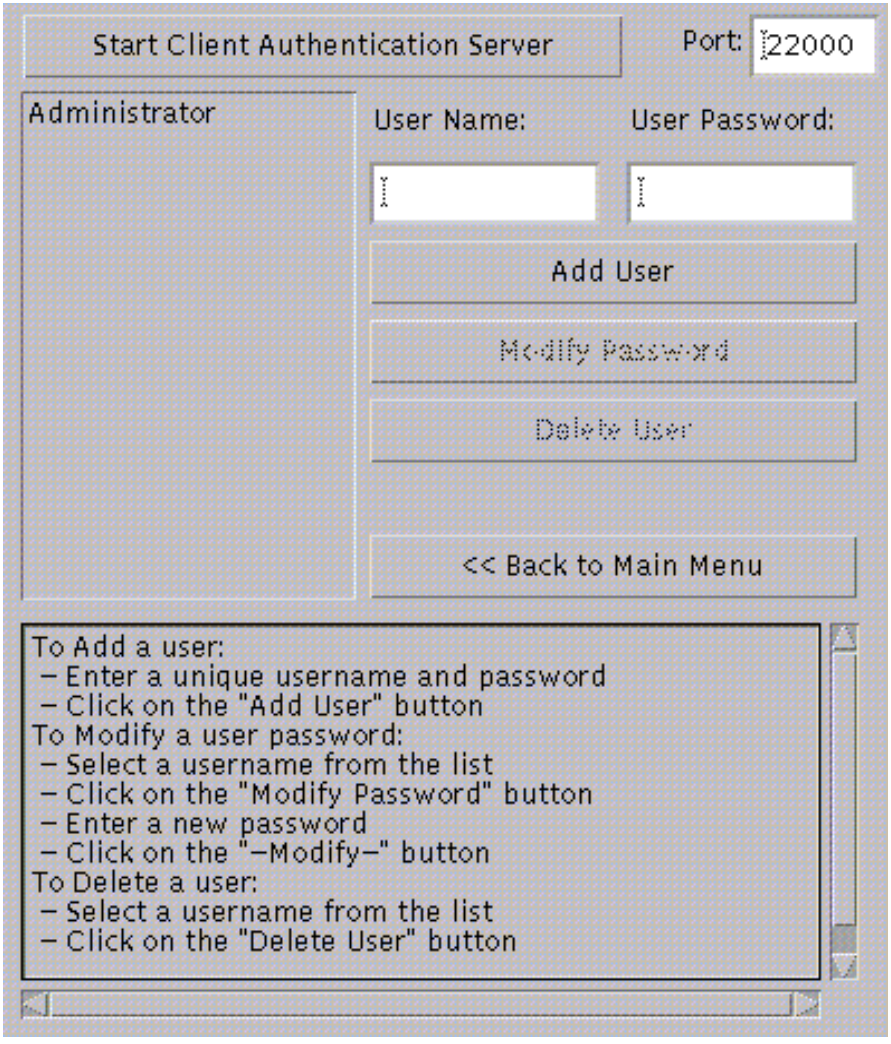
The Connections example illustrates these steps. It is located in the `examples` directory of your Java Message Queue installation.

Starting the Client Authentication Server

To start the Client Authentication Server from the Administration Console:

1. If the Administration Console is not running, launch it.
2. Click Administer Users. A panel similar to FIGURE 5-3 will be displayed.
3. On the Administer Users panel, accept the default port number or enter a specific port on which you want the Client Authentication Server to run.
4. Click Start Client Authentication Server.

FIGURE 5-3 Administer Users Panel



Alternatively, you can start the Client Authentication Server from a command prompt by typing `jmqadmin` *port number*, where *port number* is the port on which you wish to start the Client Authentication Server. For example:

```
jmqadmin 22000
```

When you use the `jmqadmin` utility to start the Client Authentication Server in this fashion, the Administration Console does not appear. Further, even if you then explicitly launch the Administration Console, you cannot administer the Client Authentication Server that is already running. If you start the Client Authentication Server from the command prompt, you must stop it the same way. For instructions on stopping the server from the command line, see "Stopping the Client Authentication Server".

Stopping the Client Authentication Server

To stop the Client Authentication Server that was started from the Administration Console:

1. From the Administration Console, click Administer Users.
2. On the Administer Users screen, click Stop Client Authentication Server.

To stop the Client Authentication Server started from the command line:

- On a Windows NT system, use the Applications panel of the Windows Task Manager to select the Administration Console entry then end it.
- In the Solaris operating environment, use `ps | grep jmqadmin` to identify the Process ID (PID) then use `kill -d` *PID* to terminate the process.

Adding a User for Authentication

To add information for a user (thus allowing the user to be authenticated by the Client Authentication Server when connecting to the router), perform the following steps:

1. If the Administration Console is not already running, launch it.
2. Click Administer Users.
3. For each user who will request access to the router for their application to run, enter a username and associated password then click Add User.

Each new user added should appear in the user list. When you are finished, click Back to Main Menu. This causes the new information to be saved.

Modifying a User's Password for Authentication

To modify a user's password, perform the following steps:

1. If the Administration Console is not already running, launch it.
2. Click Administer Users.
3. Select the user whose information you wish to modify, then click Modify Password.
4. In the User Password text field, enter a new password then click -Modify-.

When you are finished, click Back to Main Menu. This causes the new information to be saved

Deleting a User from Authentication

To delete information for a user (can no longer be authenticated by Client Authentication Server), perform the following steps:

1. If the Administration Console is not already running, launch it.
2. Click Administer Users.
3. For each user whose information you wish to remove, select the username then click Delete User.

When you are finished, click Back to Main Menu. This causes your changes to be saved.

Managing Administrative Logs

The Java Message Queue logs contain administrative events only (such as starting or stopping the Client Authentication Server), not general Java Message Queue system events (such as failure to connect to a router). For that reason, and because the logs proliferate quickly, consider your need for this information before turning logging on in the Administration Console. (By default, logging is turned off.)

Administrative logs are stored in the `admin/` subdirectory of the Java Message Queue installation. When logging is turned on, a new logfile.

Each log filename is of the form *date-time. F*or example, a log filename of `2000-3-13-15-55.log` indicates a log created on March 13, 2000 at 3:55 in the afternoon. If you start, then restart, the Log Manager within a single minute, the new log will have the same name as the old log. This results in the original file being overwritten.

An excerpt from a log might look like the following:

```
Wed Apr 05 05:54:42 PDT 2000: <JMQAdmin-securityManager> -- Started
securityManager.
Wed Apr 05 06:53:06 PDT 2000: <JMQAdmin-userManager> -- Modified user:
username1
Wed Apr 05 06:54:41 PDT 2000: <JMQAdmin-userManager> -- Added new user:
username2
Wed Apr 05 06:54:51 PDT 2000: <JMQAdmin-userManager> -- Deleted user:
username2
Wed Apr 05 06:55:00 PDT 2000: <JMQAdmin-userManager> -- Saving users...
```

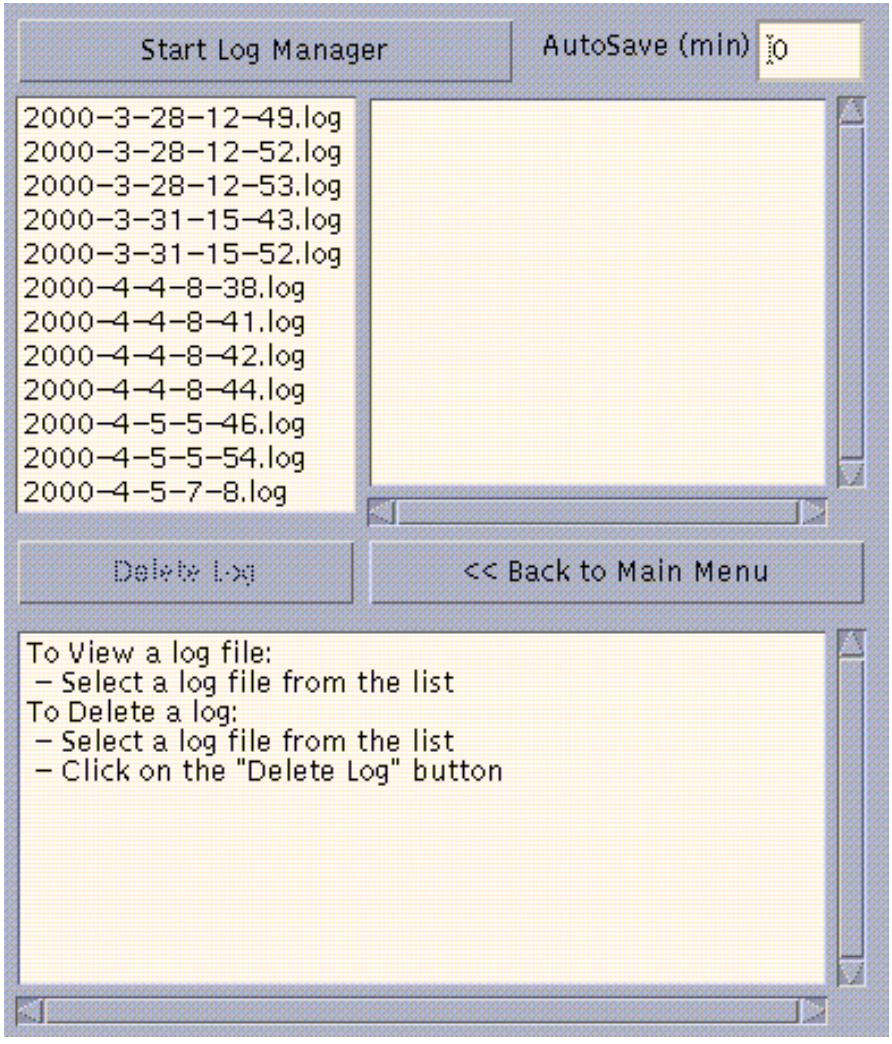
Starting and Stopping Administration Console Logging

By default, each time you start the Administration Console, logging is off. To start logging of administrative events:

1. In the Administration Console, click Administer Logs.A panel similar to FIGURE 5-4 will be displayed.
2. Click Start Logging.

Logging of administrative events will continue until you explicitly click Stop Logging or you exit the Administration Console.

FIGURE 5-4 Administer Logs Panel



Chapter 6: JMQ Router as a Windows Service

When you install the Java Message Queue product on Windows NT or 2000, you can choose to install the JMQ Router as a Windows Service. Installing the router as a Windows Service means it can be managed using the Windows Services Manager, including setting it to start automatically when the system is restarted.

If you choose not to install the JMQ Router as a Windows Service during your initial installation but later need to do so, you can use the `jmqsvcadmin` utility to accomplish this task. Conversely, the `jmqsvcadmin` utility can also be used if you initially install the JMQ Router as a Windows Service but later need to remove it.

This section provides information on starting and stopping the Router Service from the Services panel, as well as installing or removing the JMQ Router Service using the `jmqsvcadmin` utility.

Note – The `jmqsvcadmin` utility is for use on the Windows NT or 2000 platforms only. However, a similar option for automatically starting the JMQ router is provided for use in the Solaris operating environment. For more information on starting the router automatically on Solaris, see the instructions in the `/opt/SUNWjmq/etc/jmq` file.

Starting the Router Service

When the JMQ Router is installed as a Windows Service, you can use the Windows Services panel to start or stop the router service, set the service for automatic startup on system restart, and configure the automatic startup options.

Windows NT

To *manually* start the Router service on the Windows NT system:

1. From the Windows Start button, choose Settings then Control Panel.
2. In the Control Panel, double-click the Services icon.
3. In the Services list, select JMQ_Router.
4. Click Start.

To set the Router service to start *automatically* when Windows NT restarts:

1. From the Windows Start button, choose Settings then Control Panel.
2. In the Control Panel, double-click the Services icon.
3. In the Services list, select JMQ_Router.
4. Click Startup.
5. On the Startup panel, set to Automatic.

Windows 2000

To *manually* start the Router service on the Windows 2000 system:

1. From the Windows Start button, choose Programs.
2. In the Programs listing, choose Administrative Tools then Services.
3. In the Services list, select JMQ_Router.
4. Click Start.

To set the Router service to start *automatically* whenever Windows 2000 restarts:

1. From the Windows Start button, choose Programs.
2. In the Programs listing, choose Administrative Tools then Services.
3. In the Services list, select JMQ_Router.
4. Click Startup.
5. On the Startup panel, set to Automatic.

Note – Installing the JMQ Router as a Windows Service does not prevent you from starting the router (if not already started) from the command line. Starting the router process from the command line means the process continues only as long as the command window in which it is started is not closed and the user who started the process does not log off. To have the router process continue independent of those considerations, even across system restarts, you must install it as a Windows service.

The remainder of this section assumes you have already installed the router as a Windows Service. If you have not installed the router as a Windows Service and wish to do so, see the "Installing or Removing the Router Service" on page 87 for instructions.

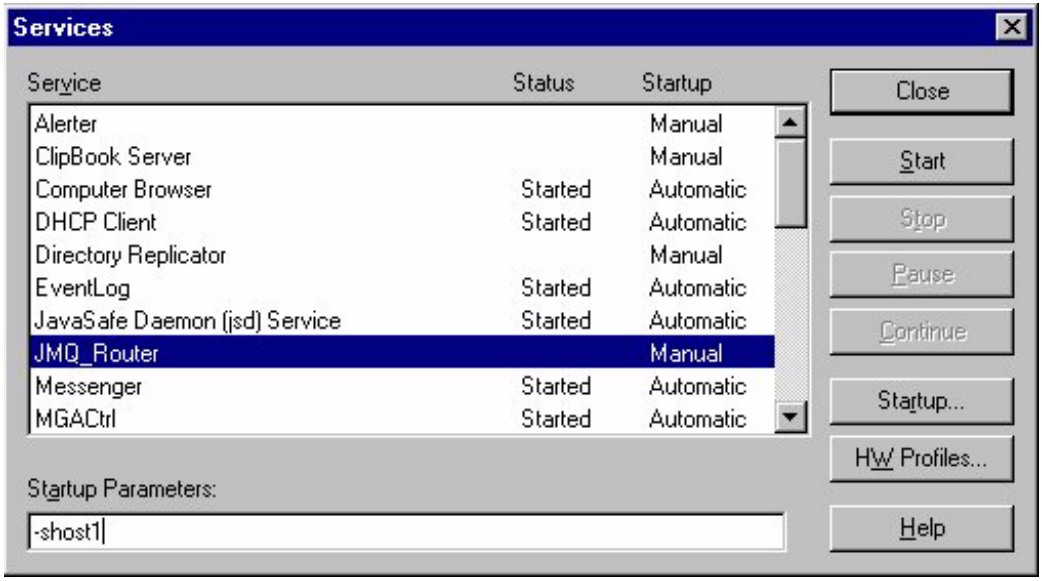
Passing Startup Options

To pass startup options to the JMQ Router service:

1. In the Windows Services panel, select JMQ_Router.
2. In the Startup Parameters textfield, type the desired startup option.
3. Click Start.

You can now close the Services panel—the running router service will not be affected. FIGURE 6-1 shows an example of passing startup options to the JMQ_Router service using the Services panel.

FIGURE 6-1 Specifying a one-time (not saved) parameter for router startup.



When you pass command-line options using the Startup Parameters textfield of the Services panel, the Windows operating system does not save the options across invocations of the Router Service. For example, if you stop and restart the router without retyping the `-shost1` Startup Parameter, the router does not automatically connect to `host1`.

Saving Startup Options Across Restarts

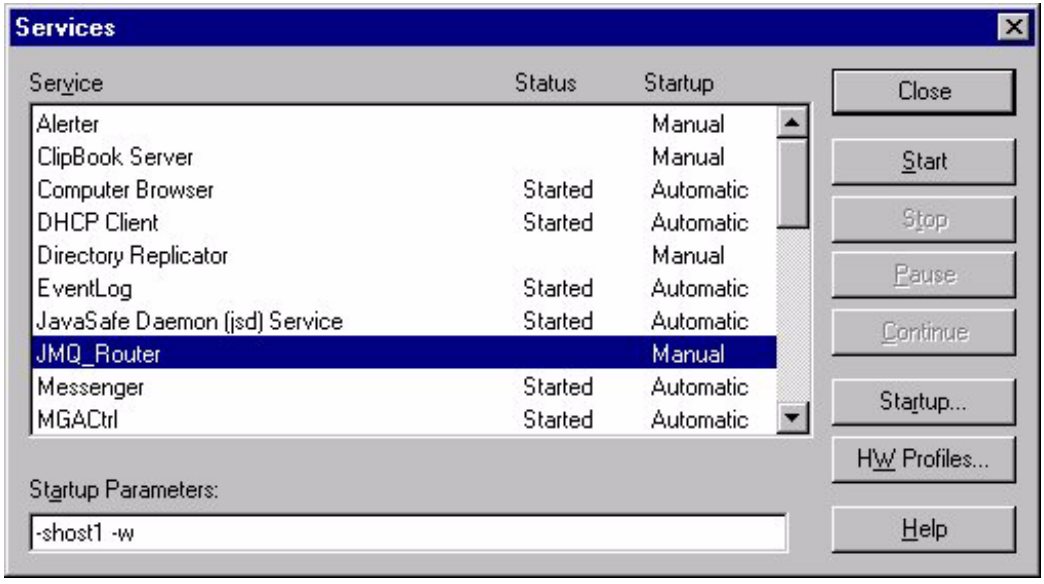
To save the startup options, such as the name of a router to which to connect, use the `-w` option. The `-w` option writes startup options to the `rtrdef.rfb` file which the JMQ Router Service reads each time it starts up. You need to be aware that options written to the `rtrdef.rfb` file will also affect routers that are started manually. For more information on the `rtrdef.rfb` file, see "JMQ Resource Files".

For example, to ensure that the JMQ Router Service always starts up connected to `host1`, enter the following options in the Startup Parameters textfield then click the Start button:

```
-shost1 -w
```

FIGURE 6-2 illustrates this startup scenario.

FIGURE 6-2 Specifying a permanent (saved) parameter for router startup.



Once you start the JMQ Router Service with this option, the router connects to `host1` each time you start it. You do not have to specify the host name again.

Installing or Removing the Router Service

The purpose of the `jmqsvcadmin` utility is to allow the JMQ Router to be installed as a Windows Service which can then be managed by the Windows Service Manager through the Services panel. The same utility is also used to remove the JMQ Router service. The `jmqsvcadmin` utility is located in the `bin` directory of the `JavaMessageQueue1.1` directory on your workstation.

Note – To run the `jmqsvcadmin` utility, you need to be logged on to your workstation with Windows `administrator` privileges. If you do not have `administrator` privileges, you will receive error responses from the utility when you attempt to install or remove the JMQ Router Service. For a description of these errors, see "jmqsvcadmin Errors" below.

---

jmqsvcadmin Syntax

The syntax of the `jmqsvcadmin` utility used for installing or removing the JMQ Router Service is:

`jmqsvcadmin` *-option*

Use this command with any of the options shown below:

- `-install`. Install the JMQ router as a Windows Service.
- `-remove`. Remove the JMQ Router Service.
- `-h`. Display help on the command-line options related to installing or removing the JMQ Router Service.

In each case, a message is displayed with the outcome of the command.

For example, the following code shows the command you would issue to install a JMQ Router as a Windows Service. You type:

```
jmqsvcadmin -install
```

The `jmqsvcadmin` utility responds:

```
Installation of service JMQ_Router successful.
```

Note – When you initially install the Java Message Queue product, you should restart your Windows workstation. Restarting the Windows operating system allows the System's PATH environment variable to be updated and for services which rely on the PATH variable to pick up the new value. The JMQ Router Service relies on the PATH environment setting for proper operation.

jmqsvcadmin Errors

If you attempt to install the JMQ Router Service when it is already installed, `jmqsvcadmin` responds:

```
Failed to install service JMQ_Router: The specified service already
exists
```

If you attempt to remove the JMQ Router Service when the Service has not been installed, `jmqsvcadmin` responds:

```
Failed to remove service JMQ_Router: The specified service does not
exist as an installed service.
```

If you attempt to install the JMQ Router Service but you are not logged on to your Windows NT or 2000 workstation with the correct privileges, `jmqsvcadmin` responds:

```
Failed to install service JMQ_Router: Access is denied.
```

If you attempt to remove the JMQ Router Service but you are not logged in on your Windows NT or 2000 workstation with the correct privileges, `jmqsvcadmin` responds:

```
Failed to remove service JMQ_Router: Access is denied
```

Chapter 7: Troubleshooting JMQ

This chapter offers solutions to problems you might encounter when running the Java Message Queue software.

Installation Problems

Although typically your installation should proceed smoothly, this section lists problems that might occur.

Unable to Connect to Router

Problem: A child process cannot connect to a router: The child process prints the JMQ banner and error messages and exits. (If you are on Windows, the child process might not open a console window, in which case, an error message is not displayed). The error message states:

```
]ERROR![ Unable to connect to router! exiting...
```

If you started the router or a utility with a `-s` command-line option, the following warnings are also displayed:

```
]Warning[ Unable to connect to router! exiting...
]Warning[ Not enough memory to alloc block
```

Solution: Ensure you are giving the correct command-line options by trying to connect to a router on the same machine. If you cannot connect to a router and you are using the correct option, the problem could be due to the network. See Chapter 2 for information on command-line options. For persistent problems, use the debugging output from the router to see what it is doing.

Verify the stack's operation with the `ircmd ping` subcommand. When using the `ping` subcommand, supply host names rather than addresses so you can verify connections among all hosts. This operation also checks that the naming service can resolve host names. You might have to use fully qualified host names such as, `granite.sun.com`.

Network System Crash

Problem: (Windows) The JMQ router and utilities crash. All network programs that use the TCP/IP stack also crash or do not work.

Solution: You might need to install or configure the TCP/IP stack.

JMQ Router and Utilities Cannot Find Host Name

Problem: The JMQ utilities cannot find a host name.

```
cmd>ping granite
No host granite
```

At startup, the following warning messages are displayed when the JMQ router and utilities are started with the `-s` option. For example:

```
]Warning[ No address available for granite
]Warning[ Error building socket address
]Warning[ Unable to connect to router! exiting...
]Warning[ Not enough memory to alloc block
```

Solution: Try using `-s` with an IP address, for example, `-s127.0.0.1`. If the IP address does not work, the problem is with the naming service (such as DNS). Add the host name to the hosts file or configure the machine to also use DNS (domain name service) or another naming services server to ensure that the name server knows the host.

Some naming services servers do not find the host if you try to use part of the host name rather than the fully qualified name (for example, `granite` as opposed to `granite.sun.com`). Try the fully qualified hostname instead of the simple host name.

JMQ Router or Utilities Crash at Startup

Problem: The JMQ router or utilities cause a generalized protection fault or crash at startup. The error message is:

```
]Warning[ Unable to resolve my own host name: granite
```

Solution: You must ensure that each machine knows its own host name. Verify that the machine can do a `ping` to its own hostname.

Network Programs Hang

Problem: The JMQ system hangs briefly during startup.

Solution: The host does not recognize the IP address. JMQ uses the host name of the workstation to look up its address at startup time. If DNS (or some other naming services server) is used, the name server must know the workstation's address.

Cannot Find DLLs

Problem: (Windows) When you attempt to start a JMQ router or utility, Windows gives the following errors:

```
File Error, Cannot find JMQ1.DLL
```

or

```
File Error, Cannot find WINSOCK.DLL
```

Solution: Ensure that the directory containing the JMQ DLL is in the correct path (`%JMQ_HOME%\bin`). You must reboot after the JMQ install to have this take effect. Ensure that the DLL location required by the TCP/IP stack is also in the search path. Frequently, network stack software does not add this directory to the path; you might have to do it manually. Note that the name of the DLL needed by the TCP/IP stack is not necessarily `winsock.dll`.

Connectivity Problems

Problem: You are having problems making a connection between two hosts.

Ensure that you can `ping` in both directions between the two hosts.

Once you know that the underlying network is configured properly, the next step is to use the command prompt program, `ircmd`. Start it up on the same subnet as your router, being sure to give it the host name of your router if it is not on the local host:

```
ircmd -n1 -sRouterMachine
```

You should obtain a prompt like `cmd>`. Type `open` to open the connection to the router and then type `hosts`. The `hosts` subcommand tells you the list of hosts that have routers that `ircmd` can talk to. You can then type `connections` *host-name* for each host. This subcommand will tell you the number and type of connection that each router has. An example is:

```
cmd>connections granite
0) (application) on granite [0xccfdd115,0x70e6]
     Type:PRIVATE,REMOTE State:ALIVE TimeDelta:0 QSize:0
1) (application) on granite [0xccfdd115,0x70ed]
     Type:PRIVATE,REMOTE State:ALIVE TimeDelta:0 QSize:0
2) (application) on granite [0xccfdd115,0x70ee]
     Type:PRIVATE,REMOTE State:ALIVE TimeDelta:0 QSize:0
```

This example shows that the router on `granite` has three connections with PIDs `0x70e6`, `0x70ed` and `0x70ee`. The number `0xccfdd115` indicates `granite`'s IP address. `PRIVATE` connections are from router to application and `PUBLIC` connections are from router to router.

Note that the connection type is REMOTE, which means that the router runs on a different host than the application. A LOCAL connection type means that router runs on the same machine as the application.

Under Windows, the router connects not to the application but to the JMQ DLL. Thus you can have three applications on one machine, but the router might show only one connection to that machine. If you find a router not connected to your network of routers / applications, you can connect to it by typing `connect Host1 to Host2`, where `Host1` is the host running a router that you are connected to, and `Host2` is the host running the router that is not yet connected.

Runtime Problems

The following section identifies problems you might have when running JMQ.

Incorrect Command Line Arguments

Problem: You have incorrectly specified command-line arguments in one of the following ways:

- The double dash (--) was not placed before the JMQ arguments causing the application to ignore the JMQ options.
- The application cannot connect to a router because the subnet (`-n`) or host (`-s`) options were not used.
- The application is using command-line options stored in the defaults file.

Solution: If there is any question about command options or inability to connect to the router, you must run the application with the `-ds1` option. This option displays all JMQ options that the application is using. Chapter 2 contains a complete listing of the options. Finally, you can examine the `rtrdef.rfb` file (in `JMQ_RESPATH` or in the current working directory) where data is stored and delete the file, if necessary.

Error Creating Semaphore

Problem: (Solaris operating environment) When trying to run a router process, the execution fails, giving an error about being unable to create a semaphore.

Solution: The number of allowed semaphores for the machine needs to be increased. Each JMQ router process requires three semaphores. To increase the number of semaphores allowed in the Solaris operating environment, add the following lines to the `/etc/system` file and reboot:

```
set semsys:seminfo_semmns=<number>
set semsys:seminfo_semmni=<number>
```

For example, to increase the number of allowed semaphores to 50, add the following lines:

```
set shmsys:shminfo_shmmax = 31457280
set shmsys:shminfo_shmmin = 1
set shmsys:shminfo_shmmni = 220
set shmsys:shminfo_shmseg = 50
set semsys:seminfo_semmns = 200
set semsys:seminfo_semmni = 70
set semsys:seminfo_semmsl = 100
```

The `ipcs` command can be used to query the status of semaphores. The `ipcs` command shows the current shared memory blocks and semaphores allocated for each user. If a user has no routers running, but still has shared memory blocks or semaphores allocated, they should be cleaned up. The router should free these blocks when the process exits, but sometimes crashes or errors interfere. The shared memory blocks and semaphores for a currently running router process are generally in sequential order. If a user has blocks or semaphores that are not in sequential order, they are probably left over from old processes.

No Such File or Directory

Problem: (Windows) JMQ fails when trying to open a resource file.

Solution: The system cannot find the defaults file you are passing to the JMQ router or another utility command with the `-f` option.

Either you are passing the wrong file name with the command (the name of the router defaults file is `rtrdef.rfb` but other commands use `defset.rfb`) or the file does not exist. Make sure that you have `JMQ_HOME` set and that the `.rfb` file is in one of the `res` directories.

JMQ Router or Utilities Crash at Runtime

Problem: The JMQ router or utilities need to be restarted at runtime.

Solution: Specify the `-eoutdev` option if you are running on Windows. This enables a better examination of any JMQ error messages. For example, on Windows, use `-ec:\error.out`. Find out whether the crash occurs at startup or later during normal processing. Check if the router and all JMQ utilities crash. Check if `ping` and other network programs work. Proceed to the correct specific problem. If you are on Windows, exit and restart Windows. This causes Windows to unload and reinitialize all DLLs of the JMQ and the TCP/IP stack.

JMQ Router Says Windows Install Directory Too Large

Problem: On Windows, the router might complain if the name of the install directory has more than eight characters or spaces.

Solution: Install the Java Message Queue product into a directory with a shorter name (such as `C:\JMQ11`).

JMQ Router or Utilities Exit Upon Startup

Problem: The system exits and the following error message is displayed.:

```
]Warning[ Unable to initialize persistent files.
```

On Windows, the router or utility exits so quickly it might look similar to a crash on startup.

Solution: Your system cannot create the directory to store persistent files (probably because of privileges).

In the Solaris operating environment:

Change the privileges on `/var/opt/SUNWjmq`.
`chmod -R a+w /var/opt/SUNWjmq`

This command grants write access to all users. However, only the user who starts the router needs to have access this directory.

On Windows:

Check and, if necessary, change privileges in the directory where JMQ is installed.

If you cannot change the privileges or changing them does not fix the problem, set the environment variable to point to a new location.

In the Solaris operating environment:

```
setenv JMQ_RESPATH ‘pwd’
```

On Windows:

```
set JMQ_RESPATH C:/tmp
```

Debugging Services

Services are DLLs (on Windows) or shared object libraries (in the Solaris operating environment). JMQ uses services to provide optional features or features that might be dependent on the configuration of router and utilities.

Service Loading Debugging

Problem: If a needed service cannot be loaded, an error message is displayed:

```
]Warning[ Unable to install service ibanner from loadlist
```

In this example, `ibanner` is the service name.

Solution: Either JMQ has not been correctly installed or JMQ cannot find the files. If this occurs, re-run the router or utility with debug level two (`-ds2`) and examine the additional error messages displayed in this mode.

Example output from this mode:

```
irouter -ds2
Kernel loading service ibanner flag 0x2
Unable to link service libibanner.so
Error linking service libibanner.so:
ld.so.1: irouter: fatal: libibanner.so: can't open file:
    errno=2
```

The router is trying to load the service `ibanner`, and cannot find the file that is the shared object `libibanner.so`.

You should verify that the shared libraries needed by the JMQ router and utilities exist.

In the Solaris operating environment:

Look for the file `lib`*service*`.so` (for example, `libbanner.so`) in the directory `/opt/SUNWjmq/lib` .

On Windows:

Use the find command to search for the *service*`.dll` file.

If the *service* file does not exist, JMQ is not installed correctly. If the file does exist, your environment variable might not be configured correctly. You should reset the environment variable.

In the Solaris operating environment:

Change the load library path to point to the directory containing the libraries:

```
setenv LD_LIBRARY_PATH /opt/SUNWjmq/lib
```

On Windows:

Add the directory containing the DLL to the path as shown:

```
set PATH=%PATH%;C:\Program Files\JavaMessageQueue1.1\bin
```

Chapter 8: Error Codes

This chapter lists JMQ error messages and error codes.

Error Messages

TABLE 8-1 lists JMQ error messages, with their error level and a short description.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`Attempted to send to closed channel`	WARNING	The system attempted to send data through a closed or disconnected connection.
`Unknown error writing socket`	WARNING	An error occurred while writing to a connection. Check to see if the connection needs to be reset.
`Unknown error writing channel`	WARNING	An error occurred while writing to a connection. Check to see if the connection needs to be reset.
`Could not generate local medium!`	FATAL	The system was unable to load information about local drivers. Check to see if the file system is installed correctly. If so, check to see if a driver is corrupted or missing.
`Could not generate default medium!`	FATAL	The system was unable to load information about local drivers. Check to see if the file system is installed correctly. If so, check to see if a driver is corrupted or missing.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`Unknown error reading channel...closing`	WARNING	A connection was shut down due to an unknown error.
`Received object with size 0...ignoring`	WARNING	An empty object was received by the router and discarded.
`Error in buffer sizes on receive package`	WARNING	The fields of an incoming message are corrupt.
`Error loading driver for medium`	WARNING	One of the drivers could not be loaded.
`Subnet is in use, unable to proceed`	FATAL	Another routing process is already listening on the subnet.
`Unable to connect to site`	FATAL	The router at that site is not responding to connection requests.
`A child process died...`	WARNING	One of the child processes terminated.
`Unable to restart...`	WARNING	The child process that died could not be restarted.
`Attempted to malloc block bigger than system maximum`	WARNING	The system attempted to allocate a memory block larger than that permitted. This might result in memory errors.
`Attempting to free CT_NULL struct!`	WARNING	An attempt was made to free a NULL structure. This might result in memory errors.
`Attempted to free invalid block!`	WARNING	An attempt was made to free an invalid memory block. This might result in memory errors.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`Attempted to free block twice!!`	FATAL	An attempt was made to free a block that has already been freed. This might result in memory errors.
`Invalid data block!`	WARNING	An invalid data block was detected (probably due to a memory corruption).
`Attempted to convert NULL size!`	WARNING	An attempt was made to resize a block with an invalid size.
`Memory corruption detected! (probably due to double-free)`	WARNING	A memory corruption was detected.
`Attempting to repair...`	WARNING	An attempt was made to repair memory corruption.
`Memory has been corrupted...unable to proceed!`	FATAL	Memory has been irretrievably corrupted. The application must exit.
`Could not listen on medium: #, subnet: #`	WARNING	The process was unable to listen for connections on the medium and subnet specified.
`Host address unknown. Connection to <hostname> failed!`	WARNING	The application could not determine the address to connect to the host.
`Unable to connect to <hostname>... retrying in 5 seconds`	WARNING or FATAL	The specifiedhost did not respond to a connect request. The application will attempt to connect again 5 seconds.
`Attempting to start router on <hostname>`	WARNING	The application is attempting to start a router on the host.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`Startup failed!`	WARNING	The application could not start a router on the host.
`Failed to connect to new router!`	WARNING	The application could not connect to the newly started router.
`No memory left for new connection!`	WARNING	The application did not have enough free memory to form the new connection.
`Error initializing new connection`	WARNING	An error occurred while initializing a new connection.
`Connection to router died!`	WARNING	The router or the connection shut down.
`Attempting to start router...`	WARNING	The application is attempting to reconnect or start the router.
`Unable to proceed...`	FATAL	The application could not continue with the existing errors.
`No connection info for open connect`	WARNING	The system could not locate connection information for an active connection.
`No connect info for indirect connect`	WARNING	The system could not locate connection information for a connection through an indirect link.
`Invalid entry in connection list!`	WARNING	An invalid entry was found in the connection list.
`No memory for startup request`	WARNING	The application does not have enough free memory to handle the startup request.
`Connection to <hostname> refused!`	WARNING	The specified host to connect to refused the connection.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`Out of memory!`	WARNING	The application is running out of memory.
`Received startup from active connection`	WARNING	A new connection was initiated from an existing connected host. This attempt will be ignored.
`Host address is NULL for host <hostname>`	WARNING	The host lookup for the specified host name failed by returning NULL.
`Unable to find host address`	WARNING	The specified host name could not be found.
`Error on received object...discarding`	WARNING	An error occurred on a received object. The object will be discarded.
`Error count exceeded threshold...shutting down link`	WARNING	The number of invalid objects received has exceeded a threshold. The link will be shut down.
`Transmit error on lost object!`	WARNING	An error occurred on sending an object, but the notification information could not be found. No notification will be sent back.
`Unable to notify!`	WARNING	A notification condition could not be reported.
`Empty element in interest queue!`	WARNING	The interest queue contained an invalid entry, probably due to removing the same entry twice.
`Unable to locate host information`	WARNING	The host information associated with an existing connection could not be found.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`Installed class <classname> is being replace by <newname>`	WARNING	An existing object class entry is being replaced with a new class with a different name. This is usually caused by a resource file conflict.
`Undefined section used as field in <classname>`	WARNING	A field of unknown type was used as a field in a class definition.
`Unknown object class! ClassID: [#,##]`	WARNING	An object of an unknown class was either read from a file, received, or created.
`Variable length element in variable field`	WARNING	Variable size fields cannot contain structures that contain variable size fields. Such a structure was set in a variable field.
`Maximum recursion level exceeded!`	WARNING	An infinite loop caused a recursion in the object distribution.
`End of file encountered in comment!`	WARNING	The end of a resource file was encountered while in a comment.
`Ignoring unknown character`	WARNING	An unknown character in the resource file was ignored.
`Error reading resfile: <filename>`	WARNING	The resource file could not be read.
`Error creating temporary resfile!`	FATAL	A temporary resource file could not be created, either due to permissions settings, or insufficient disk space.
`No resource path defined...`	FATAL	`JMQ_RESPATH` environment variable is undefined.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`No index file for directory <dirname>`	WARNING	The specified directory does not contain an index file `res.ndx`, or the permissions on the file are incorrect.
`Error reading index file: <filename>`	WARNING	The application does not have permission to read the index file in the given directory.
`Error generating resource file name!`	FATAL	The resource file name could not be generated, usually due to illegal file name length.
`Error writing resfile: <filename>`	WARNING	An error occurred while writing to the resource file.
`Error reading file label`	WARNING	An error occurred while writing the file label to a resource file.
`Resource file version mismatch`	WARNING	A resource file was detected that has a conflicting format. The file will be ignored.
`Error writing index file: <filename>`	WARNING	An error occurred while writing to the index file for the directory, due to permissions or space.
`Invalid environment selected!`	WARNING	The application selected an invalid virtual environment.
`Unable to resolve my own host name: <hostname>!`	WARNING	The application does not have local information on its own hostname. Make sure the application has its own name and address in the local hosts file or equivalent.
`Unable to execute <appname>`	WARNING	The specified application could not be executed.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
Unable to fork new process	WARNING	The fork to start a new application failed.
Error building socket address	WARNING	The parameters to the socket address creation were invalid.
Error opening socket connection	WARNING	An error occurred while opening a new connection.
A filename must be specified	WARNING	A filename must be specified for the operation.
Unable to open file <filename>	WARNING	The specified file could not be found or opened.
An error occurred while waiting	WARNING	An error occurred during a wait operation, causing the operation to fail.
line #: <message>	FATAL	Error display of line number and error condition.
Couldn't find field type: <typeid>	FATAL	The type specified for a field in an object definition is unknown.
Couldn't find property structure type: [#,##]	FATAL	The property structure type for a property definition is unknown.
Invalid description file: <filename>	WARNING	The description file is invalid.
Cannot open file <filename>	WARNING	The specified file could not be opened.
Cannot create file <filename>	WARNING	The file could not be created.
Invalid resource file: <filename>	WARNING	The resource file specified is corrupted or illegal.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`Received invalid query type: # id: #`	WARNING	An application query was of an unknown type, or cannot be handled by the application.
`A conflicting application manager was detected at address: <address>`	WARNING	Another application manager is running that conflicts. One manager will shut itself down.
`No source address for new connection`	WARNING	A physical address for the new connection could not be found.
`Received invalid status type: #`	WARNING	A status message from an application with an invalid type (that is, not the same as requested) was received.
`Error installing persistent object handle: #`	WARNING	An error occurred while registering a persistent object from disk.
`Domain ID overflow. Wrapping occured.`	WARNING	The unique identifier for domains wrapped over `0xFFFFFFF` (`IDM_MAX_DOMSEQ`).
`Overflow on address conjunction: #`	WARNING	An address conjunction overlapped the area for the address domain, causing a corruption of the domain.
`Overflow on address sequence: #`	WARNING	An address domain overlapped the area for the address conjunction causing a corruption of the conjunction.
`An unrecoverable error occured`	FATAL	An error occurred from which recovery is not possible.

TABLE 8-1 JMQ Error Messages

Message	Level	Description
`No space for spool file...closing connection`	WARNING	The system ran out of disk space and this data could not be spooled to disk.
`Illegal application instance address: <address>`	WARNING	An illegal address was received.
`Received invalid object size: #`	WARNING	An illegal object size was received.

Error Codes

TABLE 8-2l ists valid error codes within JMQ, with a short description of each.

TABLE 8-2 JMQ Error Codes

Error Code	Description
`IEC_NOERROR`	The operation succeeded.
`IEC_NOMEM`	Not enough memory for operation.
`IEC_INVSIZE`	Invalid size argument to routine.
`IEC_INVSECTYPE`	Invalid section identifier.
`IEC_INVSECPTR`	Invalid section pointer.
`IEC_INVMODE`	Mode flag is invalid.
`IEC_NOTALIGNED`	Structure is not aligned on a four-byte boundary.
`IEC_EXISTS`	The operation would overwrite an existing file or buffer.
`IEC_INVTYPE`	Invalid type argument.
`IEC_INVBLOCK`	Invalid block pointer.
`IEC_INVPARAM`	Invalid parameter passed to the routine.
`IEC_INVFLAG`	Invalid flag passed to the routine.

TABLE 8-2 JMQ Error Codes

Error Code	Description
IEC_ERRWRITE	Error writing to file.
IEC_INVDBGROUP	Invalid debug group specified in a debugging statement.
IEC_INVPORT	Invalid port number for network operation.
IEC_INVERRNO	Invalid error number specified in error display.
IEC_INVFILENAME	Invalid file name.
IEC_OPENFAILED	Could not open file or device.
IEC_CLOSEFAILED	Could not close file or device.
IEC_INVPROTOCOL	An invalid protocol was specified for a connection.
IEC_EWOULDBLK	Operation would block.
IEC_EINTR	Interrupt.
IEC_ERROR	Unknown error occurred.
IEC_NOTFOUND	Requested information was not located.
IEC_INVADDR	An invalid address was specified.
IEC_EBADF	A bad file descriptor was specified.
IEC_EPIPE	A pipe error occurred.
IEC_INTERRUPT	An interrupt occurred during the operation.
IEC_EXPIRED	The event timer expired.
IEC_EAGAIN	The operation did not complete and should be retried.
IEC_INPROGRESS	The operation is already in progress.
IEC_NORESPONSE	No response to the query activity.
IEC_NULLPARAM	One of the parameters was NULL.
IEC_DUPLICATE	A duplicate event object was received.

TABLE 8-2 JMQ Error Codes

Error Code	Description
IEC_UNDERFLOW	A mathematical underflow occurred while performing a calculation.
IEC_OVERFLOW	A mathematical overflow occurred while performing a calculation.
IEC_NOSPACE	There is not enough disk space to store a spool file or temporary file.
IEC_CLOSED	The file or connection is already closed.
IEC_MCLOSED	One of the connections in a multicast connection is already closed.
IEC_ACCESS	Access denied.
IEC_NOCONN	No more connections available on the router.

Chapter 9: System Monitoring Output

You can set command options that provide output (called print output statements) that are helpful to you when you deploy Java Message Queue.)

With print output statements, you can monitor the operations of the router and applications to determine initialization, object definitions, object distributions, and memory allocation.

Print Statements

Each print statement has associated with it:

- A group whose statements share a common output device and a common display level.
- A level that defines the print level specified on the command line or in a defaults file.

The JMQ print statement structure. composed of groups and levels thus enables multiple levels of debugging and provides increasingly detailed information.

Print Output

You can specify that the print output for each print group is either a file descriptor (usually `stdout` or `stderr`), or an output file:

- If you do not specify output, `stdout` is used and JMQ displays all print statements on the screen.
- If you specify an output file, JMQ saves the print statements to that file.

System Print Groups

System print groups provide a way of grouping output related to JMQ system events.

You can modify all system print groups through the `-ds` flag..

TABLE 9-1 provides basic information on system print groups.

TABLE 9-1 System Print Groups

System Print Group	System Print Group Number	Use
Default group	0	Prints application information.
Error group	1	Prints application errors.
SYSLVL group	2	Prints information about system, low-level operations, initializations, transport, logic and function calls, in addition to banner and copyright information.
Routing group	3	Prints router interest information.
Timing group	4	Prints timing information.
Packet group	8	Prints raw packet dumps.

System Group versus User Group Print Levels

The print level for system groups versus user groups are initialized using different default values. The system print groups for agiven application are initialized with the -ds command line option, and the use rprint groups are initialized with the user print level specified by the -du option. Both of these default to 0.

Setting Levels

JMQ generates increasingly detailed information about its current state, as print levels increase from 0 to 6. You can use the information in conjunction with `irmon` to remotely monitor and debug applications and system operations.

Print Level

Print levels 7 through 9 provide information about router packets in addition to level 6 output.

When the print level of the group is greater or equal to the debug level of the message, the message is printed to the group. If the print level number specified in a print statement is less than or equal to the current group print level, the print statement prints to the output.

Level 0

At level 0, the only output of the system print groups is the banner displayed when the application is initialized.

Level 1

The first level includes configuration and initialization details, in addition to information on services..

```
New process 0x4ced...
Cmd Flags: -c0 -ds1 -frtrdef.rfb
>>Transaction Service Ready...
>>Listening on profile default
>>Session Service Ready...
Processing service call: Service: 25 Inst: 0 Function: 1
>>Transaction Service Create
Processing service call: Service: 25 Inst: 0 Function: 5
>>Transaction Service Bind
```

This level displays notification information:

```
Sending notify type: ICS_CONN_STATUS,40000
```

This level also displays information when connections change. When a socket connection comes up, the router displays:

```
New process 0x4cf4 on host arthur...
```

The messages from the router when a socket connection goes down are as follows:

```
Channel closed...
Process 0x6e3f on host granite went away...
```

For shared memory connections, the router displays the following message when a connection comes up:

```
New process 0x4d17...
```

When a shared memory connection goes down, the router displays:

```
Process 0x6e3d went away...
```

The application program displays the following message when it opens a socket connection:

```
Connect succeeded with host granite
```

If the router itself goes down, the router displays the following message:

```
Router shutting down...
```

If the application program was using a socket connection to the router, it displays these messages when the router goes down:

```
Unable to connect to host granite using profile default
Trying again...
Unable to connect to host granite using profile default
```

For shared memory connections, no additional information is displayed by the application if the router goes down or when the application connects to the router.

Level 2

Print level 2 displays further configuration and initialization information:

```
Kernel loading service iconsrv  flag 0x2
Kernel loading service ipktsrv  rtn pointer c8c6d6  flag 0x2
Startup Parameters
---------------------------------------
    defaults file:       rtrdef.rfb
    subnet number:       0
    max connects:        0
    startup hosts:
    startup executables:
    system debug level:  2
    user debug level:    0
    timing debug level:  0
    network debug level: 0
    debug output device: stdout
    error output device: stderr
---------------------------------------
```

In addition, this level shows the first information about the interests of the process as they are added and removed. Each of the interest routines indicates whether it is a local or remote interest, the application environment, and the class of object. Remote interests show the host, hostid, object class, and the type of interest (`PUBLIC` or `PRIVATE`). Duplicate interests messages indicate a remote process already has an identical interest in that object class and properties. Local interests show the object class and information about the subroutine.

```
## NEW INTEREST: client=[0x1 0xffffff3d] IARtrInterest wt=0 handle=-195
##
```

Additionally, as objects are distributed through the system, the template of the packets and information about packet routing will appear. Lines prepended with `CONMGR` indicate information coming from the connection manager. The connection manager provides information about the adding and removing of interests and expressions, such as the following:

```
CONMGR: handle object interest type 0
```

In this case, the connection manager is reporting the addition of an interest (type 0). Type 1 represents the removal of an interest, type 2 is the addition of an expression, and type 3 is the removal of an expression.

The connection manager also displays information about packets at this level, namely the packet number, ID, and destination, as follows:

```
CONMGR: Packet:2 Dest:none
CONMGR: Packet:35547 Dest:none
CONMGR: Packet:4 Dest:[1, -193]
```

The destination is represented by [*index*, *inthandle*], where *index* is the index of the destination process in memory and *inthandle* is the interest handle. If the destination is none, there were no interests in that packet, and the packet was thrown away.

For incoming packets that have pre-routing information, the connection manager has passed a table for its associated ID to the packet router during initialization.

This table contains information that enables the packet router to determine the appropriate destinations for packets with the given ID. When these initialization steps are occurring, messages like the following are displayed:

```
Sending Route Table for [2, 1] #dest=1
    entry 0: process: 1 handle: -196 #ints: 0 fan flags: 0
    queuesizeptr 0
Sending Route Table for [2, 4] #dest=1
    entry 0: process: 1 handle: -194 #ints: 0 fan flags: 0
    queuesizeptr 0
```

Level 3

At print level 3, packet templates for distributed objects are displayed. Lines preceded by DISPATCH indicate dispatch information coming from the connection manager.

```
DISPATCH: Packet Template:
    PT_SRCDOMAIN := { PHYSICAL:arthur.modulus.com/exclass:csclient:20822
}
    PT_DESTDOMAIN := { PHYSICAL:arthur.modulus.com }
    PT_SRCMAP := { PHYSICAL:arthur.modulus.com/exclass:csclient:20822 }
    PT_DATACLASS := { IARtrInterest }
    PT_EXPRID := { [0x2, 0x2] }
```

Level 4

Print level 4 displays information when data is written to a socket, including the connection handle, the channel, the profile name, and the number of packets queued up to go out:

```
CONMGR: <#-2 CHAN:0x6440c>    PROF:default OUT:0
```

Level 5

Print level 5 adds significant more detail to the display. First, a template is added to the displayed interest information:

```
## NEW INTEREST: client=[0x1 0xffffff3f] IARtrConnStatus wt=0 handle=-
193 ##
    template:
        PT_DESTDOMAIN := { PHYSICAL:arthur.modulus.com/DEF_CLASS:19182 }
        PT_DATACLASS := { IARtrConnStatus }
```

Also, information for all received objects are displayed, including the template:

```
received object to interest -195
receiving:
    Class: IARtrInterest [0x0,0x3c] numprops = 5
        PT_SRCDOMAIN = { PHYSICAL:granite.modulus.com/DEF_CLASS:10910 }
        PT_DESTDOMAIN = { PHYSICAL:granite.modulus.com }
        PT_SRCMAP = { PHYSICAL:granite.modulus.com/DEF_CLASS:10910 }
        PT_DATACLASS = { IARtrInterest }
        PT_EXPRID = { [0x2, 0x2] }
sending:
    Class: CServerQuery [0xccfdd115,0x8e] numprops = 3
        PT_SRCDOMAIN := { PHYSICAL:granite.modulus.com/DEF_CLASS:21102 }
        PT_DESTDOMAIN := { RESOURCE:DEFAULT }
        PT_DATACLASS := { CServerQuery }
received object to interest -194
receiving:
    Class: IAServiceMsg [0x0,0x78] numprops = 4
        PT_SRCDOMAIN = { PHYSICAL:granite.modulus.com/DEF_CLASS:10910 }
        PT_DESTDOMAIN = { PHYSICAL:granite.modulus.com }
        PT_TAG = { [0x2a9e,0x8] }
        PT_DATACLASS = { IAServiceMsg }
```

This level also prints a message when packets are routed directly (no pre-routing information is provided):

```
Routing packet directly...
```

Level 6

Print level 6, additional information about services startup is displayed:

```
Constructed service/routine tag is iconsrv7
```

Packet router information is also displayed at this level. Lines preceded by PKTRTR indicate information coming from the packet router. The packet router provides information about packets, namely the packet number, ID, and destination, as follows:

```
PKTRTR: Packet:5 ID:[0, 0] Dest:free
PKTRTR: Packet:6 ID:[0, 7] Dest:entry 7
PKTRTR: packet: 2 ID: [1, 2] Dest:entry 7 died
PKTRTR: packet: 5 ID: [4, 1] Dest:[53525,-1] via [1, fffffffd]
PKTRTR: packet: 9 ID: [2, 3] Dest:[1,-196]
```

The destination is represented by [index, inthandle], where index is the index of the destination process in memory and inthandle is the interest handle. If the destination is none, that means there were no interests in that packet, and the packet was thrown away.

The first line of output above shows a packet with an ID of [0, 0]. This ID represents a packet that is being destroyed. The second line shows a normal packet. The third line shows what the packet router displays when a connection goes down. The fourth line shows a packet that is being routed to a remote slave.

The destination for such a packet is represented by [*index*, *conhandle*] via [*thread*, *inthandle*], where *index* is the index of the destination process in memory, *conhandle* is the connection handle, *thread* is the thread number of the destination process, and *inthandle* is the interest handle. The last line shows a packet with an ID of [2, 3].

This ID represents a packet that has no pre-routing information. In such cases, the connection manager must figure out the destination for this packet before it can be properly routed. The destination is calculated and then the connection manager tells the packet router directly where to send the packet.

Information is displayed whenever a new entry is added to the route table:

```
Adding route table entry for [2,1]
    entry 0: process: 1 handle: -196 #ints: 0 fan flags: 0
    queuesizeptr 0
```

Periodic information about the packet router's state is displayed for each application, as follows:

```
PKTRTR: <#0 NDX:1>    IN:0 OUT:0 FREE:2 TFREE: 46
PKTRTR: <#1 NDX:4>    IN:0 OUT:2 FREE:1 TFREE: 46
```

The first number above (for example, #0) is a sequential number maintained by the system representing the application, the index (for example, #1) is the shared memory slot number of the application, and the information listed is the number of packets coming in, going out, free, and the total number of packets free for the system.

The packet router also displays information for packets it is unable to route:

```
PKTRTR: No entry for [2, 5]...unable to route
```

Finally, packet templates for distributed objects are displayed:

```
PKTRTR: Packet Template:
    PT_SRCDOMAIN := { PHYSICAL:arthur.modulus.com/DEF_CLASS:21011 }
```

Level 7

Packet headers are displayed at level 7:

```
_____________________________________________________________________
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Header (40):
    S:128 PS:84 CT:0 ST:1 NT: 0 CS:0x00
    DC:[ccfdd115,34] PI:[2,5], PN: 0, PS: 0
    0000:00000080 00000000 00000000 00000054 00010000 00000000
        ccfdd115 00000034
    0020:00000002 00000005 00000001 00000020 00000001 0000000c
        00000014 0000000c
_____________________________________________________________________
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Level 8

At level 8, the packet properties are added to the display:

```
_____________________________________________________________________
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Header (40):
    S:276 PS:204 CT:0 ST:1 NT: 0 CS:0x00
    DC:[0,78] PI:[2,3], PN: -1, PS: 195428354
    0000:00000114 ffffffff 0ba60002 000000cc 00010000 00000000
        00000000 00000078
    0020:00000002 00000003 00000000 00000002 00000000 00000000
        00000033 00000018
Props (204):
    0000:00000044 00000004 00000030 00000014 0000000c 00000000
        00000059 00000044
    0020:00000000 0000005a 00000080 00000000 00000066 000000bc
        00000000 0000006e
    0040:000000c4 0000003c 00000000 00000000 00000001 00000020
        0000001c 00000008
    0060:ccfdd115 10000000 ccfdd115 40000015 ccfdd115 100003fc
        00000ba6 10000002
    0080:0000003c 00000000 00000000 00000001 00000020 0000001c
        00000008 ccfdd115
    00a0:10000000 ccfdd115 40000015 ccfdd115 100003fa 00000ba5
        10000000 00000ba5
    00c0:00000000 00000000 00000078 00000000 0000006e 000000ec
        00000000 00000072
_____________________________________________________________________
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Glossary

asynchronous connection An exchange of information between clients in which the client generating the event can continue immediately to its next communication without waiting for the other client to receive, process, and respond to the message.

connection A channel through which one JMS client or router sends data to another JMS client or router.

connection topology The specification of how the routers in a network are connected to one another.

connection type The type of channel through which a router communicates to another process. In JMQ, there are three types of connections: local/private, remote/private, and remote/public. Local/private means that the connection is internal to the router, and the data is internal to the application. Remote/private means that the connection is from another application, and the data is internal to the application. Remote/public means that the connection is from another application, and the data is available to a non-router application.

distributed system A system in which multiple processes—often residing on several different computers—exchange various types of information.

Domain Name Service (DNS) A distributed service used to look up an IP address, given a host name.

durable subscription An subscription that exists across client shutdowns and restarts. While a subscriber is inactive, all messages that would have been delivered to the subscriber are stored. The subscriber receives the objects when it registers the subscription again.

firewall A system designed to prevent unauthorized access to or from a private network. All network packets entering or leaving the intranet can pass through a firewall, which examines each packet and blocks those that do not meet the specified security criteria.

hosts file A file used to specify host names and their addresses.

ircmd A JMQ utility that enables you to perform simple monitoring and administrative tasks on the router. (Also called the JMQ command interface**.)**

irmon A JMQ utility that you can use to monitor and display debugging and network flow information.(Also called the JMQ monitor**.)**

JMQ Administration Console (jmqadmin) The JMQ utility used to administer JMQ client application authentication and JMQ Queues on a JMQ router.

JMQ administered objects The JMQ implementations of JMS Administered Objects that encapsulate JMQ specific behavior and enable JMS provider-independent client application development and deployment. JMQ client applications typically lookup a JMQ Administered Objects using the Java Naming and Directory Interface (JNDI) API from a Service Provider.

JMQ Configuration Tool (jmqconfig) The JMQ utility that configures and manages JMQ Administered Objects.

JMQ message service The sub-system formed by a network of one or more connected routers that provide the messaging function between JMQ client applications.

jmqsvcadmin The JMQ utility (for Windows NT and 2000 only) used to install or remove the JMQ Router Windows Service.

Java Message Queue system The system comprised of one or more JMQ client applications together with the JMQ Message Service, which client applications perform distributed tasks using JMS messages.

JMQ utilities The executables named `ircmd`, `irmon`, `jmqadmin`, `jmqconfig`, and `jmqsvcadmin`.

Java Message Queue client application A program written in the Java programming language that uses the Java Message Service (JMS) API and runs in the JMQ System. These applications are also sometimes referred to as JMS client applications or JMS clients.

JMS provider An entity that implements JMS for a messaging product.

Java Naming and Directory Interface (JNDI) An API that provides naming and directory functionality to applications written in the Java programming language. The JNDI is defined to be independent of any specific directory service implementation.

medium A physical transport. JMQ has the built-in capability of using TCP as a medium.

message A method of exchanging data between applications. A message consists of a header and a body. The header includes routing information. The body contains the application's defined content.

message selector See **selector**.

Message Time-To-Live A value that defines a message expiration time, which is the sum of the message's time-to-live and the time that it was sent.

messaging Communication between enterprise applications.

monitor application The command-line utility `irmon`, which shows debugging messages from routers running on hosts that are connected to the same router network and are using the `-dn` flag.

point-to-point communication model A communication model in which an application sends information to a specific destination that is processed by a single client. See also **publish-and-subscribe communication model**.

property A characteristic of a message that a user can set. Developers of JMQ applications can set such properties as the process that created the message, the time it was created, and how to handle its delivery.

publish-and-subscribe communication model A communication model in which an application publishes information needed by one or more client processes that subscribes to the information. See also **point-to-point communication model**.

remote router A router on a different machine than the one on which you are running. When a program needs to use a router on a remote machine, it must know the name of the host to which to connect. This can be specified using the -s option. See also **router**.

resource definition file A file that stores records of information including start-up defaults, connection configurations, and data logging.

resource index file An ASCII file named `res.ndx` that contains the names of all the installed binary resource files (`.rfb`) in the resource directory.

resources Data that pertains to a system or an application. Resources include persistent data files and messages.

`.rfb` file See **resource definition file**.

router A router forwards (or routes) messages between JMQ client applications.The router maintains Queues and Topics, and a list of subscribers to those Topics.

Refers also to the executable named `irouter` responsible for implementing the router functionality.

router network A group of routers that are connected to one another. In a direct connection, two routers are linked directly to one another.

selector A set of conditions, specified by a message consumer, that a message must satisfy so that it can be delivered to that consumer.

subnet A communications channel (for example, a port) used by all routers and applications for all JMQ communications. Routers and JMQ applications must be on the same subnet to communicate with each other.

synchronous connection The exchange of information between clients in which one client must wait for a response to a specific event before sending another message.

transaction The act of grouping a set of produced messages and a set of consumed messages into an atomic unit and marking them as either acknowledged or not.

Index

T

U

W