MQSeries® for Linux

Quick Beginnings

Version 5.2

GC34-5691-01

MQSeries® for Linux

Quick Beginnings

Version 5.2

GC34-5691-01

Note!

Before using this information and the product it supports, be sure to read the general information under "Appendix D. Notices" on page 79.

Second edition (December 2000)

This edition applies to IBM MQSeries for Linux, Version 5.2 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Tables v

Welcome to MQSeries® for Linux vii
Road map vii
Conventions vii

| **What's new in MQSeries for Linux, Version**
| **5 Release 2 . . . . ix**

Part 1. Installing MQSeries for Linux. 1

Chapter 1. Planning to install the MQSeries for Linux server. 3
Hardware requirements 3
Disk storage 3
Software requirements 4
Connectivity 4
Compilers supported for Linux applications. . 4
Options 4
Databases 5
Web servers for the MQSeries Internet Gateway 5
| Delivery. 5
Installation 5
MQSeries for Linux components and filesets . 5
MQSeries for Linux filesets 6
README file 9

Chapter 2. Installing the MQSeries for Linux, V5.2 server 11
Preparing for installation. 11
Before installation 11
| Changes to signal handling 13
| Kernel configuration 13
| Maximum open files and inodes 13
| Maximum processes 14
Installing the MQSeries for Linux server . . 15
| Installing support for Java on MQSeries. . 15
Translated books 15
| Installing the server and client on the same
| machine 16
Translated messages 16
Environment variable AMQ_INHIBIT_DLCLOSE 16
Verifying the installation of MQSeries for Linux 17
Verification procedure. 17
| Compiling user exits 23
Setting the queue manager CCSID on MQSeries for Linux 24

Chapter 3. Installing the MQSeries for Linux V5.2 client 25
Planning to install the MQSeries for Linux client 25
Hardware requirements 25
Software requirements 25
Connectivity 25
Compilers supported for Linux applications 25
Applications on Version 5.2 clients . . . 26
The installation directory. 26
Before installation 26
Installing the MQSeries for Linux client. . . 26
Translated messages 27
Verifying the installation of the MQSeries for Linux client 27
Setting up the server 27
Setting up the client 29
Putting a message on the queue 30
Getting the message from the queue . . . 30
Ending verification. 30
Uninstalling the MQSeries for Linux client. . 31

| **Chapter 4. Applying maintenance to**
| **MQSeries for Linux 33**
| Space requirements 33
| Applying the maintenance information . . . 33
| Restoring the previous service level 34

| **Chapter 5. Uninstalling MQSeries for Linux 35**

Part 2. Getting started with MQSeries 37

Chapter 6. About MQSeries 39

Introduction 39
Messages, queues, and queue managers. . . 40
Messages 40
Queues 40
Queue managers 41
MQSeries configurations 41
Channels 42
Clients and servers. 43
Clusters 43
MQSeries capabilities 44
Transactional support 44
Instrumentation events 45
Message-driven processing 46
Programming MQSeries 46

Chapter 7. Using the MQSeries command sets 47
Introducing command sets 47
Control commands. 47
MQSeries (MQSC) commands 49
PCF commands 50
Working with queue managers. 50
Creating a default queue manager . . . 50
Starting a queue manager 51
Stopping a queue manager 51
Deleting a queue manager 52
Working with MQSeries objects 52
Using the MQSC facility interactively . . 52
Ending interactive input to MQSC . . . 53
Defining a local queue 54
Displaying default object attributes . . . 54
Copying a local queue definition 55
Changing local queue attributes 56
Clearing a local queue 56
Deleting a local queue 57
Browsing queues 57

Chapter 8. Using the MQSeries Internet Gateway 59
Overview of MQSeries Internet Gateway . . 59
MQSeries Internet Gateway documentation 60

Chapter 9. Obtaining additional information 61
Hardcopy books 61
Online information. 62
Publications supplied with the product . . 62
HTML and PDF books on the World Wide Web. 64
BookManager CD-ROMs. 64
Online help 64

Part 3. Appendixes 65

Appendix A. Sample MQI programs and MQSC files 67
MQSC command file samples 67
C Program Samples 67
Supporting Databases. 68
Miscellaneous Tools 68

Appendix B. Code sets supported on MQSeries for Linux 71

Appendix C. Building applications on Linux 75
Preparing C programs 75
Compiling C programs 75
Preparing C++ programs. 76
Compilers for MQSeries for Linux . . . 76
Linking libraries 76
Selecting the libraries to match your compiler 76

Appendix D. Notices. 79
Trademarks 80

Index 83

Sending your comments to IBM 87

Tables

1. Getting started road map. vii
2. MQSeries for Linux, V5.2 server product RPM images 6
3. MQSeries for Linux V5 client RPM images 8
4. MQSeries for Linux books 61
5. MQSeries publications – file names 63
6. MQSC Command Files 67
7. Sample Programs - Source Files. . . . 67
8. Sample Programs - Databases 68
9. Miscellaneous Files 68
10. Locales and CCSIDs 71
11. C Include Files for MQSeries 75

Welcome to MQSeries® for Linux

This book describes MQSeries for Linux and explains how to plan for, install, and use the product.

Road map

Use Table 1 to find the information you need to get started with *MQSeries for Linux*.

Table 1. Getting started road map

If you want to...	**Refer to...**
Learn about system requirements for MQSeries for Linux	"Chapter 1. Planning to install the MQSeries for Linux server" on page 3
Install MQSeries for Linux	"Chapter 2. Installing the MQSeries for Linux, V5.2 server" on page 11
Install an MQSeries client	"Chapter 3. Installing the MQSeries for Linux V5.2 client" on page 25
Apply maintenance to an MQSeries client	"Chapter 4. Applying maintenance to MQSeries for Linux" on page 33
Uninstall the MQSeries for Linux client	"Chapter 5. Uninstalling MQSeries for Linux" on page 35
Read about MQSeries for Linux	"Chapter 6. About MQSeries" on page 39
Start using command sets	"Chapter 7. Using the MQSeries command sets" on page 47
Start using the Web Interface	"Chapter 8. Using the MQSeries Internet Gateway" on page 59
View or print online documentation	"Chapter 9. Obtaining additional information" on page 61
Contact IBM®	Sending your comments to IBM

Conventions

Knowing the conventions used in this book will help you use it more efficiently.

- **Boldface type** indicates the name of an item you need to select or the name of a command.
- *Italics type* indicates new terms, book titles, or variable information that must be replaced by an actual value.

Conventions

- `Monospace type` indicates an example (such as a fictitious path or file name) or text that is displayed on the screen.

What’s new in MQSeries for Linux, Version 5 Release 2

MQSeries for Linux, Version 5 Release 2 provides the following new and changed functions:

- Enhancements have been made to the performance of MQI function, channels, message logging, and application initialization and termination.
- You can now request immediate update of Object Authority Manager (OAM) data, rather than having to stop and restart the queue manager before authorization changes take effect.
- Changes have been made to the way in which OAM data is held, to improve performance.
- Support for Java™ on MQSeries is separately installable from the CD-ROM included in the MQSeries V5.2 product package. Alternatively, you can download the latest version of support for Java on MQSeries from the MQSeries Web site at:

  `http://www.ibm.com/software/mqseries/`
- Support is included for *pipelining*, which is the ability of the Message Channel Agent (MCA) to transfer messages using multiple threads.
- Channel send-exit programs can reserve space in the transmission buffer for their own use. Typically, this would be used by an exit that wanted to encrypt data and add a security key.
- Dynamic Host Configuration Protocol (DHCP) can now be used in queue manager clusters.
- Management of log files for recovery and restart has been improved.
- The area of main storage used to store information relating to a queue manager cluster can be increased dynamically. A new cluster workload-exit call (MQXCLWLN) is provided to support navigation of MQWDR, MQWQR, and MQWCR records held in dynamically increased storage.
- Minor changes to the MQSeries application programming functions have been made, including: support for MQRFH2 (the version-2 rules and formatting header); improvements to the processing of the *CodedCharSetId* field in MQSeries headers; the addition of a command-level value MQCMD_LEVEL_520; and C++ support for MQCNO Version 2 and Version 3.
- IBM WebSphere™ is supported as an XA coordinator.
- The way in which UNIX® signals are handled by MQSeries has been altered to minimize the impact on user applications.

For a complete description of new and changed function in this product, see the *MQSeries V5.2 Release Guide*.

Part 1. Installing MQSeries for Linux

Chapter 1. Planning to install the MQSeries for Linux server . . . 3
Hardware requirements . . . 3
Disk storage . . . 3
Software requirements . . . 4
Connectivity . . . 4
Compilers supported for Linux applications. . 4
Options . . . 4
Databases . . . 5
Web servers for the MQSeries Internet Gateway . . . 5
| Delivery. . . . 5
Installation . . . 5
MQSeries for Linux components and filesets . 5
MQSeries for Linux filesets . . . 6
README file . . . 9

Chapter 2. Installing the MQSeries for Linux, V5.2 server . . . 11
Preparing for installation. . . . 11
Before installation . . . 11
Creating another file system for product code . . . 13
| Changes to signal handling . . . 13
| Kernel configuration . . . 13
| Maximum open files and inodes . . . 13
| Maximum processes . . . 14
| Increasing maximum processes. . . 14
Installing the MQSeries for Linux server . . 15
| Installing support for Java on MQSeries. . 15
Translated books . . . 15
| Installing the server and client on the same machine . . . 16
Translated messages . . . 16
Environment variable AMQ_INHIBIT_DLCLOSE . . . 16
Verifying the installation of MQSeries for Linux . . . 17
Verification procedure. . . . 17
Verifying a local installation. . . . 17
Verifying a server-to-server installation 19
| Compiling user exits . . . 23
Setting the queue manager CCSID on MQSeries for Linux . . . 24

Chapter 3. Installing the MQSeries for Linux V5.2 client . . . 25
Planning to install the MQSeries for Linux client . . . 25
Hardware requirements . . . 25
Software requirements . . . 25
Connectivity . . . 25
Compilers supported for Linux applications . . . 25
Applications on Version 5.2 clients . . . 26
The installation directory. . . . 26
Before installation . . . 26
Installing the MQSeries for Linux client. . . 26
Translated messages . . . 27
Verifying the installation of the MQSeries for Linux client . . . 27
Setting up the server . . . 27
Setting up the client . . . 29
Defining a client-connection channel, using MQSERVER . . . 29
Putting a message on the queue . . . 30
Getting the message from the queue . . . 30
Ending verification. . . . 30
Uninstalling the MQSeries for Linux client. . 31

| **Chapter 4. Applying maintenance to MQSeries for Linux** . . . 33
| Space requirements . . . 33
| Applying the maintenance information . . . 33
| Restoring the previous service level . . . 34

| **Chapter 5. Uninstalling MQSeries for Linux** 35

Chapter 1. Planning to install the MQSeries for Linux server

This chapter is a summary of the requirements to run the MQSeries for Linux client, including:

- Network protocols
- Compilers
- Delivery media
- Various components of the product

The following information applies to the server environment only. For information about installing the MQSeries for Linux client, see "Chapter 3. Installing the MQSeries for Linux V5.2 client" on page 25.

Hardware requirements

MQSeries for Linux, V5.2 can be installed on any hardware that supports the Linux for Intel® operating system and that is capable of supporting MQSeries, its storage requirements, communications protocols, and applications.

Disk storage

The installation requirements depend on which components you install and how much working space you need. This, in turn, depends on the number of queues that you use, the number and size of the messages on the queues, and whether the messages are persistent. You also require archiving capacity on disk, tape, or other media.

These are the approximate storage requirements for the server:

- A minimum of 25 MB of disk space must be available for the product code and data in the file system containing the /opt directory.
- If you install the online books in HTML format, you require an additional 35 MB of storage for the books in the /opt directory.

  After installation the books are placed in the /opt/mqm/html directory.
- A minimum of 50 MB of disk space is required for data storage. To check how much disk storage you have available, use the `df` command.

Working data for MQSeries for Linux is stored by default in /var/mqm.

Note: For added confidence in the integrity of your data, you are strongly advised to put your logs onto a different physical drive from the one that you use for the queues.

Software requirements

MQSeries for Linux, V5.2 can be installed on any distribution of Linux for Intel that supports:
- Linux kernel, Version 2.2.5
- glibc, Version 2.1 or later
- pthreads, Version 0.7
- For C++ programming, libstdc++, Version 2.8.0
- For installation, Red Hat Package Manager (RPM)

MQSeries for Linux has been tested with conforming versions of the following distributions of Linux for Intel:
- Red Hat Linux
- Caldera OpenLinux
- S.u.S.E. Linux
- TurboLinux

Connectivity

MQSeries for Linux supports the TCP communications protocol. TCP is part of the Linux for Intel operating system. Any communications hardware supporting TCP can be used. In MQSeries for Linux V5.2, it is more practical to combine the use of DHCP with MQSeries queue manager clusters.

Compilers supported for Linux applications

C applications
The GNU C Compiler, 2.7.2.3 or later

C++ applications
The GNU C++ Compiler, versions 2.7.2.3, egcs-2.91.66 or 2.91.2

By default, the installed libraries are configured for the egcs-2.91.66 level of the compiler. If you want to use one of the other versions of the compiler, you must follow the extra set up instructions in “Appendix C. Building applications on Linux” on page 75.

In C, MQSeries for Linux supports both multithreaded (Posix pthread standard, draft 10 level) and single-threaded clients and local applications.

In C++, MQSeries for Linux supports single-threaded applications only.

Options

The following products can be used with MQSeries for Linux, but are not required.

Databases

- DB2® Universal Database™ V6.1

Web servers for the MQSeries Internet Gateway

- Apache Web Server, Version 1.3

Delivery

MQSeries for Linux, V5.2 is supplied on CD-ROM.

Support for Java on MQSeries is separately installable from the CD-ROM included in this product package. Alternatively, you can download support for Java on MQSeries from the MQSeries Web site, at `www.ibm.com/software/mqseries`, where the latest version of this support is always available.

Installation

MQSeries for Linux takes approximately 2 minutes to install using Red Hat Package Manager (RPM).

MQSeries for Linux components and filesets

The MQSeries for Linux server product is made up of the following components:

Runtime
Support for external applications. This does not enable you to write your own applications. The runtime component must be installed, regardless of whether you are installing the MQSeries server or the MQSeries for Linux client.

SDK Enables you to create and support your own applications; must be installed if you are installing the MQSeries server component.

Server Support for client connections; requires the runtime and SDK components to be installed. The server component is for installation only on an MQSeries server machine.

MQSeries for Linux client
The MQSeries for Linux client can be installed on the server machine, enabling you to have the MQSeries server and client on the same machine, or on a separate client machine. If you want the MQSeries for Linux client on a separate client machine, you must install it from the clients CD-ROM, not the server CD-ROM.

MQSeries Online Documentation
The books for MQSeries for Linux in HTML format. Included are:

- MQSeries Documentation

- MQSeries Internet Gateway Documentation

PDF versions of the MQSeries books are also on the CD-ROM, but are not listed as installable components.

The HTML versions of some of the MQSeries books are available in the following national languages:
- U.S. English
- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Spanish
- Simplified Chinese
- Traditional Chinese

MQSeries Internet Gateway
Provides access to MQSeries applications through HTML and CGI.

MQSeries Internet Gateway Samples
MQSeries Internet Gateway sample applications.

Man UNIX® man pages for the following commands:
- Control commands
- Message Queue Interface (MQI) commands
- MQSeries commands (MQSC)

Samples
Sample applications.

Message catalogs for these national languages:
- U.S. English
- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Spanish
- Simplified Chinese
- Traditional Chinese

MQSeries for Linux filesets

The Red Hat Package Manager (RPM) images (or filesets) that constitute the MQSeries for Linux, V5.2 server product are identified in Table 2 on page 7. To use the MQSeries server, you must install at least the runtime, server, and SDK components; other components are optional.

Table 2. MQSeries for Linux, V5.2 server product RPM images

RPM image name	Description
MQSeriesRuntime-5.2.0-0.i386.rpm	Runtime package; must be installed.
MQSeriesServer-5.2.0-0.i386.rpm	Server package; must be installed.
MQSeriesSDK-5.2.0-0.i386.rpm	Package that supports application development; must be installed.
MQSeriesClient-5.2.0-0.i386.rpm	Client package.
MQSeriesSamples-5.2.0-0.i386.rpm	MQSeries sample applications; must be installed if samples are used to verify installation.
MQSeriesMan-5.2.0-0.i386.rpm	UNIX man pages for MQSeries commands.
MQSeriesMsg_fr-5.2.0-0.i386.rpm	Message catalogs (French)
MQSeriesMsg_it-5.2.0-0.i386.rpm	Message catalogs (Italian)
MQSeriesMsg_ja-5.2.0-0.i386.rpm	Message catalogs (Japanese)
MQSeriesMsg_ko-5.2.0-0.i386.rpm	Message catalogs (Korean)
MQSeriesMsg_pt-5.2.0-0.i386.rpm	Message catalogs (Brazilian Portuguese)
MQSeriesMsg_Zh_CN-5.2.0-0.i386.rpm	Message catalogs (Simplified Chinese)
MQSeriesMsg_Zh_TW-5.2.0-0.i386.rpm	Message catalogs (Traditional Chinese)
MQSeriesMsg_de-5.2.0-0.i386.rpm	Message catalogs (German)
MQSeriesMsg_es-5.2.0-0.i386.rpm	Message catalogs (Spanish)
MQSeriesHtml_en-5.2.0-0.i386.rpm	MQSeries HTML-format books (U.S. English)
MQSeriesHtml_Zh_CN-5.2.0-0.i386.rpm	MQSeries HTML-format books (Simplified Chinese)
MQSeriesHtml_Zh_TW-5.2.0-0.i386.rpm	MQSeries HTML-format books (Traditional Chinese)
MQSeriesHtml_de-5.2.0-0.i386.rpm	MQSeries HTML-format books (German)
MQSeriesHtml_es-5.2.0-0.i386.rpm	MQSeries HTML-format books (Spanish)
MQSeriesHtml_fr-5.2.0-0.i386.rpm	MQSeries HTML-format books (French)
MQSeriesHtml_it-5.2.0-0.i386.rpm	MQSeries HTML-format books (Italian)
MQSeriesHtml_ja-5.2.0-0.i386.rpm	MQSeries HTML-format books (Japanese)
MQSeriesHtml_ko-5.2.0-0.i386.rpm	MQSeries HTML-format books (Korean)
MQSeriesHtml_pt-5.2.0-0.i386.rpm	MQSeries HTML-format books (Brazilian Portuguese)
MQSeriesDMQ_runtime-5.2.0-0.i386.rpm	MQSeries Internet Gateway.

Table 2. MQSeries for Linux, V5.2 server product RPM images (continued)

RPM image name	Description
MQSeriesDMQ_samples-5.2.0-0.i386.rpm	MQSeries Internet Gateway sample applications.
MQSeriesDMQDoc_base-5.2.0-0.i386.rpm ges	MQSeries Internet Gateway HTML-format books (base)
MQSeriesDMQDoc_de-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (German)
MQSeriesDMQDoc_en-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (U.S. English)
MQSeriesDMQDoc_es-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (Spanish)
MQSeriesDMQDoc_fr-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (French)
MQSeriesDMQDoc_it-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (Italian)
MQSeriesDMQDoc_ja-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (Japanese)
MQSeriesDMQDoc_ko-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (Korean)
MQSeriesDMQDoc_Zh_CN-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (Simplified Chinese)
MQSeriesDMQDoc_Zh_TW-5.2.0-0.i386.rpm	MQSeries Internet Gateway HTML-format books (Traditional Chinese)

The Red Hat Package Manager (RPM) images (or filesets) that you can install on an MQSeries for Linux client machine are identified in Table 3. To use the MQSeries for Linux client, you must install at least the runtime and client components; other components are optional.

Table 3. MQSeries for Linux V5 client RPM images

RPM image name	Description
MQSeriesRuntime-5.2.0-0.i386.rpm	Runtime package; must be installed.
MQSeriesClient-5.2.0-0.i386.rpm	Client package; must be installed.
MQSeriesSDK-5.2.0-0.i386.rpm	Package that supports application development.
MQSeriesSamples-5.2.0-0.i386.rpm	MQSeries sample applications; must be installed if samples are used to verify installation.
MQSeriesMsg_fr-5.2.0-0.i386.rpm	Message catalogs (French)

Table 3. MQSeries for Linux V5 client RPM images (continued)

RPM image name	Description
MQSeriesMsg_it-5.2.0-0.i386.rpm	Message catalogs (Italian)
MQSeriesMsg_ja-5.2.0-0.i386.rpm	Message catalogs (Japanese)
MQSeriesMsg_ko-5.2.0-0.i386.rpm	Message catalogs (Korean)
MQSeriesMsg_pt-5.2.0-0.i386.rpm	Message catalogs (Brazilian Portuguese)
MQSeriesMsg_Zh_CN-5.2.0-0.i386.rpm	Message catalogs (Simplified Chinese)
MQSeriesMsg_Zh_TW-5.2.0-0.i386.rpm	Message catalogs (Traditional Chinese)
MQSeriesMsg_de-5.2.0-0.i386.rpm	Message catalogs (German)
MQSeriesMsg_es-5.2.0-0.i386.rpm	Message catalogs (Spanish)

README file

Before starting to install MQSeries for Linux, V5.2, please see the README file in the MQSeries for Linux package for latest information.

Chapter 2. Installing the MQSeries for Linux, V5.2 server

This chapter tells you how to install MQSeries for Linux, V5.2 and how to verify that your installation has been successful.

The MQSeries product is installed into the /opt/mqm directory. This cannot be changed. However, if you do not have enough space in the /opt/mqm file system, follow the procedure given in “Creating another file system for product code” on page 13.

See the following section for any instructions specific to your distribution.

Preparing for installation

This section guides you through some of the steps you must perform before you install MQSeries for Linux.

Before installation

Before you can install MQSeries for Linux you:

- If you have installed the MQSeries for Linux V5.1 Technology Release, you must uninstall it before attempting to install MQSeries for Linux V5.2 on the same machine.
- Must check the README file for latest information.
- Are recommended to create a group with the name mqm. (If you do not create group mqm before installing, the group is created automatically, with default values, during the installation.)
- Must add root to the mqm group if you want to use root as an MQSeries administrator ID.
- Must create the mqm group and mqm user ID before installing the MQSeries for Linux product if you are using Caldera Linux.
- Are recommended to create a user ID with the name mqm. (If you do not create user ID mqm before installing, the user ID is created automatically, with default values, during the installation.)
- May install, using RPM, a package that provides the /bin/sh files, or override the dependency checking using the `--nodeps` option on the command line for the install.
- Are recommended to create and mount either a /var/mqm file system, or /var/mqm, /var/mqm/log, and /var/mqm/errors file systems.

  If you are creating separate file systems, you should allow a minimum of:
  - 30 MB of storage for /var/mqm
  - 2 MB of storage for /var/mqm/errors

– 20 MB of storage for /var/mqm/log

If you are using a single file system, use the sum of these figures as a guide.

Notes:

1. The /var/mqm file system should be large enough to contain all the messages, on all the queue managers, on this system.
2. If you create separate partitions, the following directories must be on a local file system:
   – /var/mqm
   – /var/mqm/log

   You can choose to NFS mount the /var/mqm/errors and /var/mqm/trace directories to conserve space on your local system.
3. The size of the log file depends upon the log settings that you use. The size recommended is for circular logging using the default settings. For further information on log sizes see the *MQSeries System Administration* book.
4. The inode information associated with the files in directory /var/mqm and in the directories below /var/mqm/qmgrs is used to identify uniquely the attributes of each queue manager. To ensure that this information remains unique, the maximum number of inodes in the file systems containing these directories should not exceed 8288608 (2ˆ23).

   By default, the number of inodes in a file system is proportional to its size: for an ext2 file system that has 1 inode per 4 KB, file systems up to 24 GB do not exceed the 8288608 limit. If you need a file system larger than 24 GB, you must either create multiple file systems or increase the size of each inode when the file system is created.

   To discover the number of inodes associated with the file system containing the /var/mqm directory, enter:

   ```
   df -i /var/mqm
   ```

   The maximum number of inodes is shown in column two.

After installation, the user ID mqm owns the directories and files that contain the resources associated with the product. This group and user must be defined for any machine on which the MQSeries software is to be installed, whether the machine is a client or a server machine.

If you want to run any administration commands, for example, **crtmqm** (create queue manager) or **strmqm** (start queue manager), your user ID must be a member of group mqm.

For stand-alone machines, you can create the new user and group IDs locally. For machines administered in a network information services (NIS) domain, you can create the user and group IDs on the NIS master server machine.

Creating another file system for product code

If you do not want to have the product code installed in the /opt/mqm file system (for example, that file system might be too small to contain the product) you can do one of two things:

1. Create a new file system and mount it as /opt/mqm.
2. Create a new directory anywhere on your machine that is large enough to contain the product, and create a symbolic link from /opt/mqm to this new directory. For example:

   ```
   mkdir /bigdisk/mqm
   ln -s /bigdisk/mqm /opt/mqm
   ```

Notes:

1. Whichever of these options you pick, you must do it before installing the product code.
2. The file system into which the code is installed can be a remote network device, for example NFS, provided that the mount options are defined on that device to allow setuid programs – including root access – to be run.
3. Distributions that do not use the RPM installer by default can generate an error when you install MQSeries.

Changes to signal handling

In MQSeries for Linux V5.2, the way in which signals are handled has been changed. These changes, and their effects on your existing applications, are described in the *MQSeries V5.2 Release Guide*.

Kernel configuration

MQSeries for Linux makes use of shared memory; it is probable that the default kernel configuration will be adequate. If the system is heavily loaded you might need to increase the maximum shared memory segment size (shmmax). To change shmmax, enter the command:

```
echo 65536000 > /proc/sys/kernel/shmmax
```

It is recommended that you add this command to a startup script in /etc/rc.d/...

Maximum open files and inodes

If the system is heavily loaded, you might need to increase the maximum possible number of open files and inodes, as follows:

```
echo 32768 > /proc/sys/fs/file-max
echo 65536 > /proc/sys/fs/inode-max
```

You are recommended to add these commands to a startup script in /etc/rc.d/....

Maximum processes

By default, the maximum number of processes in a Linux system is 512, and the maximum number of processes per user is 256. For most applications, these values are sufficient. However, because MQSeries uses threaded applications, and because the Linux implementation of threads results in a new process being created for each thread, these values can be exceeded quickly, especially on a heavily loaded system. To increase these values, you must first rebuild the Linux kernel (the documentation for your Linux distribution contains detailed instructions for rebuilding the kernel) and then enable the pam_limits module and alter the values /etc/security/limits.conf. To enable the pam_limits module you have to edit the files in etc/pam.d and add the line:

```
session     required     /lib/security/pam_limits.so
```

This line should be added to all files named after methods of logging on to the system, that is `login`, `rlogin`, `rexec`, `rsh`, and `su`.

You must now add the following lines to /etc/security/limits.conf:

```
@mqm soft nproc 4090
@mqm hard nproc 4090
@mqm soft nofile 10240
@mqm hard nofile 10240
```

This sets the limits to the values required for the users in the mqm group.

Note: You are strongly recommended to read the information about rebuilding the kernel in your Linux documentation. Failure to build the kernel correctly could make your Linux system unusable.

The steps involved in increasing the maximum number of processes are summarized here.

Increasing maximum processes

This procedure assumes that:
- The kernel is installed under /usr/src/linux
- The GNU C compiler is installed.

1. Log in as root.
2. Increase the maximum number of processes by changing the NR_TASKS value in the file /usr/src/linux/include/linux/tasks.h. The maximum value of NR_TASKS is 4090.

The maximum number of processes per user is set automatically to half the NR_TASKS value, though you can set this explicitly by changing the value of MAX_TASKS_PER_USER.

3. Change to the /usr/src/linux directory.
4. Run the kernel configuration utility by entering:
   ```
   make menuconfig
   ```
   Further information about running the configuration utility is in the Linux documentation.
5. Build the kernel by entering:
   ```
   make dep clean bzImage
   ```
6. Install the kernel by entering:
   ```
   make install
   ```
   Alternatively, you can install the kernel manually. See the Linux documentation for more information.
7. Build and install any configured modules by entering:
   ```
   make modules modules_install
   ```
8. Verify the lilo configuration and restart the system.

Installing the MQSeries for Linux server

To install any MQSeries for Linux server component, use the rpm –i command. For example, to install the mandatory RPM images for the MQSeries for Linux server, enter:

```
rpm -i MQSeriesRuntime-5.2.0-0.i386.rpm
rpm -i MQSeriesSDK-5.2.0-0.i386.rpm
rpm -i MQSeriesServer-5.2.0-0.i386.rpm
```

Installing support for Java on MQSeries

Support for Java on MQSeries is now separately available on CD-ROM and from the MQSeries Web site at:

www.ibm.com/software/mqseries

Translated books

If you install the HTML-format documentation (the MQSeriesPub_ images), be aware that some books are not available in languages other than U.S. English, therefore some hypertext links between books may not work. To overcome this you must install a complete set of books in U.S. English in addition to those in your national language. See "Online information" on page 62 for more information about hypertext linking between translated books.

Note also that, regardless of national language, the appropriate base image (MQSeriesDMQDoc_base-5.2.0-0.i386.rpm) must be installed.

Installing the server

If you install the MQSeries for Linux client on the server machine, you need to configure MQSeries communication (that is, to define an MQI channel) between server and client. For information about this task, see “Chapter 3. Installing the MQSeries for Linux V5.2 client” on page 25 or the *MQSeries Clients* book.

Installing the server and client on the same machine

To install an MQSeries for Linux client on the server machine, use the MQSeries server CD-ROM. Choose the client install option on the server CD-ROM to install the client code on the server machine. Do not use the clients CD-ROM.

You might install components from the MQSeries clients CD-ROM on a machine and then later want to install the MQSeries server component on the same machine. If so, you must first remove from the machine any of the components that were installed from the MQSeries clients CD-ROM. You can then use the MQSeries server CD-ROM to install the server, client, and any other components that you need. You cannot install the server on a machine that already has other components installed from the MQSeries clients CD-ROM.

For information about installing the client on a different machine from the server, see “Chapter 3. Installing the MQSeries for Linux V5.2 client” on page 25 .

Translated messages

Messages in U.S. English are always available. If you have installed the message catalogs for another of the languages that is supported by MQSeries for Linux, V5.2, you must ensure that your NLSPATH environment variable includes the appropriate directory.

For example:

```
export LANG=de
export NLSPATH=/usr/share/locale/%L/LC_MESSAGES/%N
```

Environment variable AMQ_INHIBIT_DLCLOSE

You should currently set the environment variable `AMQ_INHIBIT_DLCLOSE=TRUE` globally to overcome a known problem with base Linux. Details of the known problem can be found at `www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl/full/1738`. The effect of setting the environment variable is to prevent MQSeries from calling the `dlclose()` function to unload shared libraries which were dynamically loaded using `dlopen()` within MQSeries libraries. The typical effect of not setting this environment variable is a segmentation violation shortly after an MQDISC from a threaded process (either an MQSeries internal process, or a user application). This is a temporary workaround and will not be required when a fix to the base Linux problem is available.

Verifying the installation of MQSeries for Linux

This section describes how to verify that MQSeries for Linux has been correctly installed and configured. You do this by following the steps outlined in “Verification procedure”.

If you want to verify a communications link between multiple MQSeries installations (for example between two servers or between a client and a server), you must ensure that TCP has been installed and configured on both machines.

You can verify a local installation (which has no communications links with other MQSeries installations) without having TCP installed and configured.

Verification procedure

You can verify an MQSeries installation at three levels:

- A local (stand-alone) installation, involving no communication links to other MQSeries machines
- A server-to-server installation, involving communication links with other MQSeries servers
- A client/server installation, involving communication links between a server machine and an MQSeries client

Verification of local and server-to-server installations is described in “Verifying a local installation”, and in “Verifying a server-to-server installation” on page 19. For information on verifying a client/server installation, see “Chapter 3. Installing the MQSeries for Linux V5.2 client” on page 25.

Verifying a local installation

Follow these steps to install and test a simple configuration of one queue manager and one queue, using sample applications to put a message onto the queue and to read the message from the queue:

1. Install MQSeries for Linux on the workstation.
2. Create a default queue manager (in this example called venus.queue.manager).
   - At the command prompt type:
     ```
     crtmqm -q venus.queue.manager
     ```
   - Press Enter.

Messages are displayed telling you that the queue manager has been created, and that the default MQSeries objects have been created.

3. Start the default queue manager by typing the following command and pressing Enter:

   ```
   strmqm
   ```

   A message tells you when the queue manager has started.

4. Enable MQSC commands by typing the following command and pressing Enter:

   ```
   runmqsc
   ```

   Note: MQSC has started when the following message is displayed: `Starting MQSeries Commands.`

   MQSC has no command prompt.

5. Define a local queue (in this example, called ORANGE.QUEUE).
   - Type the following and press Enter:

     ```
     define qlocal (orange.queue)
     ```

     Note: Any text entered in MQSC in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. This means that if you create a queue with the name orange.queue, you must remember to refer to it in any commands outside MQSC as ORANGE.QUEUE.

     The message `MQSeries queue created` is displayed when the queue has been created.

   You have now defined:
   - A default queue manager called venus.queue.manager
   - A queue called ORANGE.QUEUE

6. Stop MQSC by typing `end`, and pressing Enter.

   The following message is displayed:

   ```
   One MQSC commands read.
   No commands have a syntax error.
   All valid MQSC commands were processed.
   ```

7. The command prompt is now displayed again.

To test the queue and queue manager, use the samples amqsput (to put a message on the queue) and amqsget (to get the message from the queue):

1. Change into the directory /opt/mqm/samp/bin
2. To put a message on the queue, type the following command and press Enter:

```
amqsput ORANGE.QUEUE
```

The following message is displayed:

```
Sample amqsput0 start
target queue is ORANGE.QUEUE
```

3. Type some message text and press Enter twice.

   The following message is displayed:

   ```
   Sample amqsput0 end
   ```

   Your message is now on the queue and the command prompt is displayed again.
4. If you are not already in directory /opt/mqm/samp/bin, change to it now.
5. To get the message from the queue, type the following command and press Enter:

   ```
   amqsget ORANGE.QUEUE
   ```

   The sample program starts, your message is displayed, the sample ends, and the command prompt is displayed again.

   The verification is complete.

Verifying a server-to-server installation

The steps involved in verifying a server-to-server installation are more complex, because the communications link between the two machines must be checked.

Follow these steps to set up two workstations, one as a sender and one as a receiver.

Sender workstation:

1. Create a default queue manager called saturn.queue.manager.
   - At a command prompt in a window, type:

     ```
     crtmqm -q saturn.queue.manager
     ```
   - Press Enter.

     Messages are displayed telling you that the queue manager has been created, and that the default MQSeries objects have been created.
2. Start the queue manager.
   - Type the following command and press Enter:

     ```
     strmqm
     ```

     A message tells you when the queue manager has started.

3. Enable MQSeries Commands (MQSC) by typing the following command and pressing Enter:

   ```
   runmqsc
   ```

   Note: MQSC has started when the following message is displayed: `Starting MQSeries Commands`.

   MQSC has no command prompt.
4. Define a local queue to be used as a transmission queue, called TRANSMIT1.QUEUE.
   - Type the following command and press Enter:

     ```
     define qlocal (transmit1.queue) usage (xmitq)
     ```

     The message `MQSeries queue created` is displayed when the queue has been created.
5. Create a local definition of the remote queue.

   ```
   define qremote (local.def.of.remote.queue) rname (orange.queue) +
   rqmname ('venus.queue.manager') xmitq (transmit1.queue)
   ```

   Note: The RNAME parameter specifies the name of the queue on the remote machine to which the message is being sent. Therefore, the name specified by the RNAME parameter (ORANGE.QUEUE) must be the same as the name of the queue to which the message is being sent (ORANGE.QUEUE on the receiver workstation).
6. Define a sender channel:

   ```
   define channel (first.channel) chltype (sdr) conname (9.20.11.182) +
   xmitq (transmit1.queue) trptype (tcp)
   ```

   where 9.20.11.182 is the TCP address of the receiver workstation (note that this example is TCP specific).

   You have now defined the following objects:
   - A default queue manager called saturn.queue.manager
   - A transmission queue called TRANSMIT1.QUEUE
   - A remote queue called LOCAL.DEF.OF.REMOTE.QUEUE
   - A sender channel called FIRST.CHANNEL
7. Stop MQSC by typing `end`, and pressing Enter.
8. Configure a listener to start the MQI channels.

   You can use either the standard UNIX TCP listener or the MQSeries listener to start the MQI channels. For performance reasons, the MQSeries listener is likely to be the better choice.

   To configure the MQSeries listener to start MQI channels, use the **runmqlsr** control command:

```
runmqlsr -m queue manager -t tcp -p port number
```

To run the job in the background, add an ampersand character (&) to the end of the command.

To configure the inetd daemon to start the MQI channels, you must be logged in as a superuser, or as root.

a. Edit the file /etc/services. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries       1414/tcp      # MQSeries channel listener
   ```

   1414 is the default port number. If you are using a different port, specify its number instead.

b. Edit the file /etc/inetd.conf. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries stream tcp nowait mqm /opt/mqm/bin/amqcrsta amqcrsta
   ```

   Note: If saturn.queue.manager is not the default queue manager on this workstation, you need to add -m saturn.queue.manager to the end of this line.

c. Run the command:

   ```
   kill -1 processid of inetd daemon
   ```

Now set up the receiver workstation.

Receiver workstation:

Note: You must be logged in as a superuser, or as root, to perform step 1 to step 3.

1. Edit the file /etc/services. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries       1414/tcp      # MQSeries channel listener
   ```

2. Edit the file /etc/inetd.conf. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries stream tcp nowait mqm /opt/mqm/bin/amqcrsta amqcrsta
   ```

   Note: If you are not creating venus.queue.manager as the default queue manager on this workstation, you need to add -m venus.queue.manager to the end of this line.

3. Run the command:

   ```
   kill -1 processid of inetd daemon
   ```

4. Create a default queue manager (in this example called venus.queue.manager).
   - At the command prompt, type:

```
crtmqm -q venus.queue.manager
```

- Press Enter.

  Messages are displayed telling you that the queue manager has been created, and that the default MQSeries objects have been created.

5. Start the queue manager.
   - Type the following command and press Enter:

     ```
     strmqm
     ```

     A message tells you when the queue manager has started.
6. Enable MQSC by typing the following command:

   ```
   runmqsc
   ```

   Note: MQSC has started when the following message is displayed: `Starting MQSeries Commands`.

   MQSC has no command prompt.
7. Define a local queue (in this example, called ORANGE.QUEUE).
   - Type the following command and press Enter:

     ```
     define qlocal (orange.queue)
     ```

     The message `MQSeries queue created` is displayed when the queue has been created.
8. Create a receiver channel:

   ```
   define channel (first.channel) chltype (rcvr) trptype (tcp)
   ```

   You have now defined the following objects:
   - A default queue manager called venus.queue.manager
   - A queue called ORANGE.QUEUE
   - A receiver channel called FIRST.CHANNEL
9. Stop MQSC by pressing Ctrl-D or typing `end` and pressing Enter.

Establishing communication between the workstations:

1. If the queue managers on the two workstations have been stopped for any reason, restart them now (using the **strmqm** command).
2. On the sender workstation start the sender channel:
   ```
   runmqchl -c FIRST.CHANNEL -m saturn.queue.manager &
   ```
   The receiver channel on the receiver workstation is started automatically when the sender channel starts.
3. On the sender workstation, use the amqsput sample program to send a message to the queue on the receiver workstation:
   ```
   amqsput LOCAL.DEF.OF.REMOTE.QUEUE
   ```
   Note: You put the message to the local definition of the remote queue, which in turn specifies the name of the remote queue.
4. Type the text of the message and press Enter twice.
5. On the receiver workstation, use the amqsget sample program to get the message from the queue:
   ```
   amqsget ORANGE.QUEUE
   ```
   The message is displayed.

   The verification is complete.

Compiling user exits

User exit programs are supported by MQSeries for Linux.

- For information about data-conversion exits, see the *MQSeries Application Programming Guide.*
- For information about channel exits, see the *MQSeries Intercommunication* book.
- For information about cluster-workload exits, see the *MQSeries Queue Manager Clusters* book.

Compile a user exit program for a nonthreaded environment as follows:

```
gcc -shared -lmqm -ldl -o exit <exit>.c
```

Compile a user exit program for a multithreaded environment as follows:

```
gcc -D_REENTRANT -shared -lmqm_r -lpthread -ldl -o exit_r <exit>.c
```

Setting the queue manager CCSID on MQSeries for Linux

The coded character set identifier (CCSID) is fixed when the queue manager is created. The CCSID used is the one for the code set of the locale that you are using to run the **crtmqm** command.

For example:

```
export LC_ALL=en_US
```

uses the code set ISO8859-1 and sets a CCSID of 819.

To modify an existing queue manager CCSID, follow this procedure:

1. Record the existing queue manager CCSID, using the MQSeries (MQSC) command:
   ```
   DISplay QMGR CCSID
   ```
2. Change the CCSID to the new CCSID, with the MQSC command:
   ```
   ALTer QMGR CCSID
   ```
3. Stop the queue manager.
4. Restart the queue manager and any channels it uses.

See “Appendix B. Code sets supported on MQSeries for Linux” on page 71 for further information about supported code sets.

Chapter 3. Installing the MQSeries for Linux V5.2 client

This chapter describes how to:

- Plan for the installation of the MQSeries for Linux client.
- Install the MQSeries for Linux client on a machine other than the MQSeries for Linux server machine.
- Verify communication between the MQSeries for Linux client and an MQSeries server.

Planning to install the MQSeries for Linux client

This section identifies:

- The hardware and software required by the MQSeries for Linux client
- Tasks you must perform before installing the MQSeries for Linux client

Hardware requirements

The MQSeries for Linux client can be installed on any Linux for Intel desktop or server computer with a minimum of 25 MB of disk space.

Software requirements

The MQSeries for Linux client can be installed on any distribution of Linux for Intel that supports:

- Linux kernel, Version 2.2.5
- | glibc, Version 2.1 or later
- pthreads, Version 0.7
- For installation, Red Hat Package Manager (RPM)

Connectivity

The MQSeries for Linux client supports the TCP communications protocol. TCP is part of the Linux for Intel operating system. Any communications hardware supporting TCP can be used.

Compilers supported for Linux applications

C applications

The GNU C Compiler, 2.7.2.3 or later

C++ applications

| The GNU C++ Compiler, versions 2.7.2.3, egcs-2.91.66 or 2.91.2

| By default, the installed libraries are configured for the egcs-2.91.66
| level of the compiler. If you want to use one of the other versions of
| the compiler, you must follow the extra set up instructions in
| “Appendix C. Building applications on Linux” on page 75.

Applications on Version 5.2 clients

A Version 5 client can connect to all queue managers that support client attach. Note, however, that you cannot use features and structures specific to Version 5.2 products in your client application when connecting to a non-Version 5.2 queue manager.

The installation directory

The MQSeries for Linux client is installed into the /opt/mqm directory. This cannot be changed. However, if you do not have enough space in the /opt/mqm file system, follow the procedure given in “Creating another file system for product code” on page 13.

Before installation

Before you can install the MQSeries for Linux client, you:

- Must check the README file for latest information.
- Are recommended to create a group with the name mqm. (If you do not create group mqm before installing, the group is created automatically, with default values, during the installation.)
- Are recommended to create a user ID with the name mqm. (If you do not create user ID mqm before installing, the user ID is created automatically, with default values, during the installation.)
- Are recommended to create and mount either a /var/mqm file system, or /var/mqm, /var/mqm/log, and /var/mqm/errors file systems.

Notes:

1. If you create separate partitions, the directory /var/mqm must be on a local file system.

   You can NFS mount the /var/mqm/errors and /var/mqm/trace directories to conserve space on your local system.
2. After installation, the user ID mqm owns the directories and files that contain the resources associated with the product. This group and user must be defined for any machine on which the MQSeries software is to be installed, whether the machine is a client or a server machine.
3. For stand-alone machines, you can create the new user and group IDs locally (or allow them to be created automatically during the installation). For machines administered in a network information services (NIS) domain, you must create the user and group IDs on the NIS master server machine before beginning the installation.

Installing the MQSeries for Linux client

This section describes how to install the MQSeries for Linux client components.

To install any of the MQSeries for Linux V5.2 client components, use the `rpm –i` command. For example, to install the mandatory RPM images for the MQSeries for Linux client, enter:

```
rpm -i MQSeriesRuntime-5.2.0-0.i386.rpm
rpm -i MQSeriesClient-5.2.0-0.i386.rpm
```

Translated messages

Messages in U.S. English are always available. If you have installed the message catalogs for another of the languages that is supported by MQSeries for Linux, V5.2, you must ensure that your NLSPATH environment variable includes the appropriate directory.

For example:

```
export LANG=de
export NLSPATH=/usr/lib/locale/%L/LC_MESSAGES/%N
```

Verifying the installation of the MQSeries for Linux client

You can verify that your client installation has completed successfully, and that the communication link between client and server is working, using the supplied sample PUT and GET programs.

The verification process comprises the following tasks:
1. Setting up the server
2. Setting up the client
3. Putting a message on the queue
4. Getting the message from the queue
5. Ending verification

These instructions assume that:
- The full MQSeries product has been installed on a UNIX server machine. If you are connecting the MQSeries for Linux client to a non UNIX MQSeries server, please see the *MQSeries Clients* book for verification instructions.
- The MQSeries for Linux client software has been installed on the client machine.
- TCP/IP is configured and initialized on both the server and the client machines.

Note: MQSeries object definitions are case sensitive. You must type the examples exactly as shown.

Setting up the server

On the server, create a directory to hold working files (called mqverify, for example), and make this the current directory. Follow these steps to set up the server workstation:

Verifying the installation

1. Create a default queue manager called saturn.queue.manager by entering the following command at the command prompt:
   ```
   crtmqm -q saturn.queue.manager
   ```
2. Start the queue manager by entering the following command:
   ```
   strmqm
   ```
3. Start MQSeries commands (MQSC) by entering the following command:
   ```
   runmqsc
   ```
   MQSC does not provide a prompt, but should respond with the message:
   ```
   Starting MQSeries Commands
   ```
4. Create a local queue called QUEUE1 by entering the following command:
   ```
   DEFINE QLOCAL(QUEUE1)
   ```
5. Create a server-connection channel by entering the following command:
   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ')
   ```
6. Stop MQSC by typing `end` and pressing Enter.
7. Configure a listener to start the MQI channels.

   You can use either the standard UNIX TCP listener or the MQSeries listener to start the MQI channels. For performance reasons, the MQSeries listener is likely to be the better choice.

   To configure the MQSeries listener to start MQI channels, use the **runmqlsr** control command:
   ```
    runmqlsr -m queue manager -t tcp -p port number
   ```
   To run the job in the background, add an ampersand character (&) to the end of the command.

   To configure the inetd daemon to start the MQI channels, you must be logged in as a superuser, or as root.

   a. Edit the file /etc/services. If you do not have the following line in that file, add it as shown:
      ```
      MQSeries       1414/tcp      # MQSeries channel listener
      ```
      1414 is the default port number. If you are using a different port, specify its number instead.

   b. Edit the file /etc/inetd.conf. If you do not have the following line in that file, add it as shown:
      ```
      MQSeries stream tcp nowait mqm /opt/mqm/bin/amqcrsta amqcrsta
      ```
      Note: If saturn.queue.manager is not the default queue manager on this workstation, you need to add -m saturn.queue.manager to the end of this line.

   c. Run the command:

```
kill -1 processid of inetd daemon
```

Setting up the client

When an MQSeries application is run on the MQSeries client, the information it requires is:

- The name of the MQI channel
- The communications protocol
- The address of the server

You provide this information by defining a client-connection channel. The name used for this channel must be the name used for the server-connection channel defined on the server. In this example the MQSERVER environment variable is used to define the client-connection channel. This is the simplest way, although not the only one.

Before starting, type `ping server-address` (where server-address is the TCP/IP host name of the server) to confirm that your MQSeries client and server TCP/IP sessions have been initialized. You can use the network address, in the format n.n.n.n, in the **ping** command instead of the host name.

If the **ping** command fails, check that your TCP/IP software is correctly configured and has been started.

Defining a client-connection channel, using MQSERVER

Create a client-connection channel by setting the MQSERVER environment variable, as follows:

```
export MQSERVER=CHANNEL1/TCP/'server-address(port)'
```

where:

CHANNEL1
Is the name of the server-connection channel already defined on the server.

TCP Is the communications protocol.

server-address
Is the TCP/IP host name of the server.

(port) Is optional and is the TCP/IP port number the server is listening on. If you do not give a port number MQSeries uses the one specified in the QM.INI file. If no value is specified in the QM.INI file, MQSeries uses the port number identified in the TCP/IP services file for the service name MQSeries. If this entry in the services file does not exist, a default value of 1414 is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

Putting a message on the queue

On the MQSeries client workstation, put a message on the queue using the amqsputc sample program:

1. From a command prompt window, change to the directory (/opt/mqm/samp/bin) containing the sample program amqsputc.
2. Enter the following command:

   ```
   amqsputc QUEUE1 saturn.queue.manager
   ```

   where saturn.queue.manager is the name of the queue manager on the server.

   The following message is displayed:

   ```
   Sample AMQSPUT0 start
   target qname is QUEUE1
   ```
3. Type some message text and press Enter twice.

   The following message is displayed:

   ```
   Sample AMQSPUT0 end
   ```

The message is now on the queue.

Getting the message from the queue

On the MQSeries client workstation, get a message from the queue using the amqsgetc sample program:

- From the directory containing the sample programs, enter the following command:

  ```
  amqsgetc QUEUE1 saturn.queue.manager
  ```

  Where saturn.queue.manager is the name of the queue manager on the server.

The message is removed from the queue and displayed.

Ending verification

The verification process is now complete.

You can stop the queue manager on the server by typing:

```
endmqm saturn.queue.manager
```

If you want to delete the queue manager on the server, enter

```
dltmqm saturn.queue.manager
```

Uninstalling the MQSeries for Linux client

To uninstall the MQSeries for Linux client, issue the `rpm` command followed by the –e option and the name of the package you want to uninstall.

For example, to uninstall the MQSeries for Linux client and runtime components, enter the command

```
rpm -e MQSeriesClient-5.2.0-0
rpm -e MQSeriesRuntime-5.2.0-0
```

To list all the installed MQSeries components, enter

```
rpm -q -a | grep "^MQSeries"
```

If you have made any modifications in the /opt/mqm or /var/mqm directories since you carried out the installation you will need to manually delete the files in those directories.

Chapter 4. Applying maintenance to MQSeries for Linux

This chapter tells you how to apply maintenance to MQSeries for Linux.

Maintenance updates in the form of a Program Temporary Fix (PTF) are supplied on CD-ROM. They can also be downloaded from `www.ibm.com/software/ts/mqseries`.

You must not have any queue managers running when you install maintenance on MQSeries for Linux. To end all running queue managers, enter the command:

```
endmqm -i QMgrName
```

Check that the queue manager has ended then check that the message queue manager is not running by entering the command:

```
endmqm -p QMgrName
```

Alternatively, enter the command:

```
ps -ef | grep amq
```

Where | is the pipe symbol.

Check that there are no amq files listed and ignore any file names that start amqi.

Space requirements

A PTF requires hard disk space for installation. In addition, the installation process requires up to an identical amount of disk space to save the previous level. For example, a 16 MB PTF requires up to 32 MB of space.

Files updated by a PTF installation are kept in a backup directory. This allows a PTF to be removed and the previous level to be automatically restored. Previous levels are stored in the directory /opt/mqm/ptf_backup. Do not delete or move these directories and files.

Applying the maintenance information

To apply the maintenance information in the PTF:

1. Put the CD in the CD-ROM drive on your PC.
2. On the command line, type:

   ```
   rpm -i /mnt/cdrom/mq_linux/mqm/rpmname
   ```

3. Press Enter.

For further information on using rpm to install software packages, see your Linux documentation.

Restoring the previous service level

To restore the previous service level:

1. Log in as root or use the command `su`.
2. Enter the `rpm` command to remove the latest PTF from your system.(For example, to remove a service update, enter the command:

   ```
   rpm -e MQSeriesServer-5.2.0-2
   ```

Details of the rpm command can be found in your Linux documentation, or by entering the `man rpm` command.

Chapter 5. Uninstalling MQSeries for Linux

Before you can uninstall MQSeries for Linux 5.2, you must find out the names of the MQSeries packages currently installed on your system. To do this use the command

```
rpm -q -a | grep MQ
```

to list all of the packages with their version information. To list all the installed packages using only their MQSeries names, use the command

```
rpm -q -a --queryformat "%{NAME}\n" | grep MQ
```

The packages are listed following the command. To remove a package, use the command

```
rpm -e MQSeriesServer
```

where MQSeriesServer is the name of a package.

Some of the installed MQSeries packages are dependent on others. The `rpm` command will not remove a package if others are dependent on it. For this reason you must uninstall the MQSeries packages in such an order that each one you uninstall has no dependencies from other packages.

To list all of the MQSeries packages on which a named package depends, use the command

```
rpm -q --requires MQSeriesServer
```

where MQSeriesServer is the package name. The list generated by this command displays those MQSeries packages on which the named package depends. This means that you can remove the named package but not those displayed in the list. Were you to remove those they would damage the package you identified in the command.

The list below shows one such permissible order where the `rpm` command at the top of the list is the first to be executed and the one at the bottom is the last. Of course, only those lines from the list that correspond to installed packages should be executed.

Note: You can make carrying out these commands easier by placing them in a shell script.

```
rpm  -e  MQSeriesMan
rpm  -e  MQSeriesHtml_Zh_TW
rpm  -e  MQSeriesHtml_Zh_CN
rpm  -e  MQSeriesHtml_pt
rpm  -e  MQSeriesHtml_ko
rpm  -e  MQSeriesHtml_ja
rpm  -e  MQSeriesHtml_it
rpm  -e  MQSeriesHtml_fr
rpm  -e  MQSeriesHtml_es
rpm  -e  MQSeriesHtml_en
rpm  -e  MQSeriesHtml_de
rpm  -e  MQSeriesHtml_base
rpm  -e  MQSeriesDMQDoc_Zh_TW
rpm  -e  MQSeriesDMQDoc_Zh_CN
rpm  -e  MQSeriesDMQDoc_pt
rpm  -e  MQSeriesDMQDoc_ko
rpm  -e  MQSeriesDMQDoc_ja
rpm  -e  MQSeriesDMQDoc_it
rpm  -e  MQSeriesDMQDoc_fr
rpm  -e  MQSeriesDMQDoc_de
rpm  -e  MQSeriesDMQDoc_es
rpm  -e  MQSeriesDMQDoc_en
rpm  -e  MQSeriesDMQSamples
rpm  -e  MQSeriesDMQRuntime
rpm  -e  MQSeriesMsg_Zh_TW
rpm  -e  MQSeriesMsg_Zh_CN
rpm  -e  MQSeriesMsg_pt
rpm  -e  MQSeriesMsg_ko
rpm  -e  MQSeriesMsg_ja
rpm  -e  MQSeriesMsg_it
rpm  -e  MQSeriesMsg_fr
rpm  -e  MQSeriesMsg_es
rpm  -e  MQSeriesMsg_de
rpm  -e  MQSeriesSamples
rpm  -e  MQSeriesClient
rpm  -e  MQSeriesServer
rpm  -e  MQSeriesSDK
rpm  -e  MQSeriesRuntime
```

Alternatively, you can write a more complex script in Perl. This could query rpm to discover the names of those installed MQSeries packages with the prefix MQSeries and then build a data structure that lists the dependencies of each of these packages using the command

```
rpm -q -requires
```

You can use the data structure to find packages upon which others have no dependencies and begin uninstalling those until none remain.

To uninstall all of the files at the same time, enter the command:

```
rpm -qa | grep "MQSeries" | xargs rpm -e
```

Part 2. Getting started with MQSeries

Chapter 6. About MQSeries 39
Introduction 39
Messages, queues, and queue managers. . . 40
Messages 40
Queues 40
Queue managers 41
MQSeries configurations 41
Channels 42
Clients and servers. 43
Clusters 43
MQSeries capabilities 44
Transactional support 44
Instrumentation events 45
Message-driven processing 46
Programming MQSeries 46

Chapter 7. Using the MQSeries command sets 47
Introducing command sets 47
Control commands. 47
Using control commands. 47
MQSeries (MQSC) commands 49
Running MQSC commands 50
PCF commands 50
Working with queue managers. 50
Creating a default queue manager . . . 50
Starting a queue manager 51
Stopping a queue manager 51
Quiesced shutdown 51
Immediate shutdown 51
Preemptive shutdown. 52
Deleting a queue manager 52
Working with MQSeries objects 52
Using the MQSC facility interactively . . 52
Feedback from MQSC commands . . . 53
Ending interactive input to MQSC . . . 53
Defining a local queue 54
Displaying default object attributes . . . 54
Copying a local queue definition 55
Changing local queue attributes 56
Clearing a local queue 56
Deleting a local queue 57
Browsing queues 57

Chapter 8. Using the MQSeries Internet Gateway 59
Overview of MQSeries Internet Gateway . . 59
MQSeries Internet Gateway documentation 60

Chapter 9. Obtaining additional information 61
Hardcopy books 61
Online information. 62
Publications supplied with the product . . 62
HTML 62
PDF. 63
HTML and PDF books on the World Wide Web. 64
BookManager CD-ROMs. 64
Online help 64

Chapter 6. About MQSeries

This chapter introduces IBM MQSeries. It describes its basic functions and its relationships with operating systems, applications, and other middleware products.

Introduction

MQSeries is a communications system that provides assured, asynchronous, once-only delivery of data across a broad range of hardware and software platforms.

These characteristics make MQSeries the ideal infrastructure for application-to-application communication, and make it an appropriate solution whether the applications run on the same machine or on different machines that are separated by one or more networks.

MQSeries supports all the important communication protocols and even provides routes between networks that use different protocols. MQSeries bridges and gateway products allow easy access (with little or no programming) to many existing systems and application environments—for example, Lotus® Notes™, Web browsers, Java applets, and many others.

The assured delivery capability reflects the many functions built in to MQSeries to ensure that data is not lost because of failures in the underlying system or network infrastructure. Assured delivery enables MQSeries to form the backbone of critical communication systems and to be entrusted with delivering high-value data. There are also options that allow you to select a less robust quality of service, where this is appropriate. For example, there might be circumstances where you might prefer faster delivery with less emphasis on assured delivery.

The asynchronous processing support in MQSeries means that the exchange of data between the sending and receiving applications is time independent. This allows the sending and receiving applications to be decoupled so that the sender can continue processing, without having to wait for the receiver to acknowledge that it has received the data. In fact, the target application does not even have to be running when the data is sent. Likewise, the entire network path between the sender and receiver may not need to be available when the data is in transit.

Once-only delivery of data is a vital consideration, particularly in financial and business applications where duplicate requests to move large sums of money from one account to another are precisely what you do not want to happen!

Messages, queues, and queue managers

The three fundamental concepts in MQSeries that you need to understand are:
- Messages
- Queues
- Queue managers

Messages

A *message* is a string of bytes that has meaning to the applications that use it. Messages are used for transferring data from one application to another (or to different parts of the same application). The applications can be running on the same platform, or on different platforms.

MQSeries messages have two parts; the *application data* and a *message descriptor*. The content and structure of the application data is defined by the application programs that use the data. The message descriptor identifies the message and contains other control information, such as the type of message and the priority assigned to the message by the sending application.

Queues

A *queue* is a data structure in which messages are stored. The messages may be put on, or got from, the queue by applications or by a queue manager as part of its normal operation.

Queues exist independently of the applications that use them. A queue can exist in main storage (if it is temporary), on disk or similar auxiliary storage (if it must be kept in case of recovery), or in both places (if it is currently being used, and must also be kept for recovery). Each queue belongs to a *queue manager,* which is responsible for maintaining it. The queue manager puts the messages it receives onto the appropriate queue.

Queues can exist either in your local system, in which case they are called *local queues,* or at another queue manager, in which case they are called *remote queues.*

Applications send to, and receive messages from, queues. For example, one application can put a message on a queue, and another application can get the message from the same queue.

Each queue has *queue attributes* that determine what happens when applications reference the queue. The attributes indicate:

- Whether applications can retrieve messages from the queue (get enabled)
- Whether applications can put messages onto the queue (put enabled)
- Whether access to the queue is exclusive to one application or shared between applications
- The maximum number of messages that can be stored on the queue at the same time (maximum queue depth)
- The maximum size of messages that can be put on the queue (maximum message size)

Queue managers

A queue manager provides queuing services to applications, and manages the queues that belong to it. It ensures that:

- Object attributes are changed according to the details received.
- Special events (such as instrumentation events or triggering) are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application. The application is informed if this cannot be done, and an appropriate reason code is given.

Each queue belongs to a single queue manager and is said to be a *local queue* to that queue manager. The queue manager to which an application is connected is said to be the local queue manager for that application. For the application, the queues that belong to its local queue manager are local queues. A *remote queue* is a queue that belongs to another queue manager. A *remote queue manager* is any queue manager other than the local queue manager. A remote queue manager may exist on a remote machine across the network or it may exist on the same machine as the local queue manager. MQSeries supports multiple queue managers on the same machine.

MQSeries configurations

In the simplest configurations, MQSeries is installed on a machine and a single queue manager is created. This queue manager then allows you to define queues. Local applications can then use these queues to exchange messages.

Communication by applications with queues managed by another queue manager requires *message channels* to be defined. It is not necessary to define a channel directly to the target queue manager and it is often appropriate to define one only to the next hop (that is, an intermediate queue manager). Message channels available from that queue manager will be used to deliver the message to the target queue manager (or even to a subsequent hop).

More complex configurations can be created using a client-server structure. The MQSeries product can act as an MQSeries server to MQSeries clients. The clients and server do not need to be on the same platform. MQSeries supports a broad range of client platforms. The MQSeries products typically include clients for a variety of platforms. Additional MQSeries clients are available from the MQSeries Web site.

In a client-server configuration, the MQSeries server provides messaging and queuing services to the clients, as well as to any local applications. The clients are connected to the server through dedicated channels (known as *client channels*) for clients. This is a cost-effective deployment method because a server can support hundreds of clients with only a single copy of the MQSeries server product. However, the client channel must be continuously available whenever the MQSeries applications on the client are running. This contrasts with the message channels, which need not be continuously available to support MQSeries applications running on the server.

See “Channels” for more information.

MQSeries also supports the concept of *clusters* to simplify setup and operation. A cluster is a named collection of queue managers and any one queue manager can belong to none, one, or several such clusters. The queue managers in a cluster can exist on the same or different machines.

There are two major benefits from the use of clusters:

1. Communication between members of a cluster is greatly simplified, particularly because the channels required for exchanging messages are automatically defined and created as needed.
2. Some or all of the queues of participating queue managers can be defined as being cluster queues, which has the effect of making them automatically known and available to all other queue managers in the cluster.

See “Clusters” on page 43 for more information.

Channels

A channel provides a communication path to a queue manager. There are two types of channel: message channels and MQI channels.

A *message channel* provides a communication path between two queue managers on the same, or different, platforms. The message channel is used for transmitting messages from one queue manager to another, and shields the application programs from the complexities of the underlying networking protocols. A message channel can transmit messages in one direction only. Two message channels are required if two-way communication is required between two queue managers.

A *client channel* (also known as an *MQI channel*) connects an MQSeries client to a queue manager on a server machine and is bidirectional.

If you want to read more information about channels and how MQSeries uses them to communicate across the systems in your network, see the *MQSeries Intercommunication* book.

Clients and servers

MQSeries supports client-server configurations for MQSeries applications.

An *MQSeries client* is a part of the MQSeries product that is installed on a machine to accept MQSeries calls from applications and pass them to an *MQSeries server* machine. There they are processed by a queue manager. Typically, the client and server reside on different machines, but they can also exist on the same machine.

An *MQSeries server* is a queue manager that provides queuing services to one or more clients. All the MQSeries objects (for example, queues) exist only on the queue manager machine (that is, on the MQSeries server machine). A server can support local MQSeries applications as well.

The difference between an MQSeries server and an ordinary queue manager is that the MQSeries server can support MQSeries clients, and each MQSeries client application has a dedicated communication link with the MQSeries server.

For more information about client support, see the *MQSeries Clients* book.

Clusters

A cluster is a named collection of queue managers.

Clusters require that at least one of the queue managers in the cluster be defined as a *repository* (that is, a place where the shared cluster information can be held). More typically, two or more such repositories are usually designated to provide continued availability in the case of system failure. MQSeries makes sure that the information in the repositories is synchronized.

When a queue is defined as a cluster queue, it can be regarded as a public queue in that it is freely available to other queue managers in the cluster. This contrasts with noncluster queues, which are accessible only when a local definition of them is available. Thus, a noncluster queue has the characteristics of a private queue, accessible only to those queue managers that have been configured to know about them.

Public queues with the same name in the same cluster are regarded as equivalent. If a message is sent to that queue name, MQSeries (by default) sends it to any one of the instances, using a load-balancing algorithm. If you do not want this to happen, you can use the queue manager and queue name in the address, thus forcing the message to be delivered to a specific queue manager. Alternatively, you can replace the load-balancing routine with a different implementation. This is typical of MQSeries, in that there are many examples of where standard behavior can be changed by implementing user code in exits designed for this purpose.

You can read a full explanation in the *MQSeries Queue Manager Clusters* book.

MQSeries capabilities

MQSeries can be used to create many different types of solutions. Some exploit the platform support, or the bridge and gateway capabilities, to connect existing systems in an integrated way or to allow new applications to extract information from, or interchange information with, existing systems. Other solutions support business application servers, where a central pool of MQSeries applications can manage work sent across networks. Complex routing of information for workflow scenarios can be supported. Publish/subscribe or “send and forget” are other application scenarios that use different message flows. Load balancing and hot-standby systems can be built using the power and flexibility of MQSeries, which includes specific functions to support many of these diverse scenarios.

See the *MQSeries Application Programming Guide* for more information about writing MQSeries applications.

Transactional support

An application program may need to group a set of updates into a *unit of work*. Such updates are usually logically related and must all be successful for data integrity to be preserved. Data integrity would be lost if one update in the group succeeded while another failed. MQSeries supports transactional messaging.

A unit of work *commits* when it completes successfully. At this point all updates made within that unit of work are made permanent and irreversible. Alternatively, all updates are *backed out* if the unit of work fails. *Syncpoint coordination* is the process by which a unit of work is either committed or backed out with integrity.

A *local* unit of work is one in which the only resources updated are those of the MQSeries queue manager. Here, syncpoint coordination is provided by the queue manager itself, using a single-phase commit process.

A *global* unit of work is one in which resources belonging to other resource managers, such as XA-compliant databases, are also updated. Here, a two-phase commit procedure must be used and the unit of work may be coordinated by the queue manager itself, or externally by another XA-compliant transaction manager such as IBM CICS®, IBM Transaction Server, IBM TXSeries™, Transarc Encina, or BEA Tuxedo.

When the queue manager coordinates global units of work itself it becomes possible to integrate database updates within MQSeries units of work. That is, a mixed MQSeries and SQL application can be written, and commands can be used to commit or roll back the changes to the queues and databases together.

The queue manager achieves this using a two-phase commit protocol. When a unit of work is to be committed, the queue manager first asks each participating database manager whether it is prepared to commit its updates. Only if all of the participants, including the queue manager itself, are prepared to commit, are all of the queue and database updates committed. If any participant cannot prepare its updates, the unit of work is backed out instead.

Full recovery support is provided if the queue manager loses contact with any of the database managers during the commit protocol. If a database manager becomes unavailable while it is in doubt (that is, it has been called to prepare but has yet to receive a commit or backout decision), the queue manager remembers the outcome of the unit of work until it has been successfully delivered. Similarly, if the queue manager terminates with incomplete commit operations outstanding, these are remembered when the queue manager restarts.

Instrumentation events

You can use MQSeries instrumentation events to monitor the operation of queue managers.

Instrumentation events cause special messages, called *event messages*, to be generated whenever the queue manager detects a predefined set of conditions. For example, a *Queue Full* event message is generated if: Queue Full events are enabled for a specified queue; an application issues an MQPUT call to put a message on that queue; and the call fails because the queue is full.

Other conditions that can give rise to instrumentation events include:
- A predefined limit for the number of messages on a queue being reached
- A queue not being serviced within a specified time
- A channel instance being started or stopped

If you define your event queues as remote queues, you can put all the event queues on a single queue manager (for those nodes that support instrumentation events). You can then use the events generated to monitor a network of queue managers from a single node.

MQSeries instrumentation events are categorized as follows:

Queue manager events
These are related to the definitions of resources within queue managers. For example, if an application attempts to open a queue but the associated user ID is not authorized to perform that operation, a queue manager event is generated.

Performance events
These are notifications that a threshold condition has been reached by a resource. For example, a queue depth limit has been reached or, following an MQGET request, a queue has not been serviced within a predefined period of time.

Channel events
These are reported by channels as a result of conditions detected during their operation. For example, a channel event is generated when a channel instance is stopped.

Message-driven processing

When they arrive on a queue, messages can automatically start an application, using a mechanism known as *triggering*. If necessary, the application can be stopped when the message or messages have been processed.

Programming MQSeries

MQSeries applications can be developed using a variety of programming languages and styles. Procedural and object-oriented programming is supported, depending on the MQSeries platform, using, for example, Visual Basic®, C, C++, Java, COBOL, and PL/I.

MQSeries function is logically divided into what is normally required by applications (such as putting messages on a queue) and what is necessary for administration (such as changing queue or queue manager definitions). Application function is known as the *MQI* (message queue interface). Administration function is known as the *MQAI* (message queuing administration interface). Applications can mix MQI and MQAI functionality, as required.

The administration functions can be implemented in two ways:

1. Most often, using MQAI language bindings
2. Sending messages to administration queues, to achieve the same results as with the MQAI, using programmable command formats (PCFs)

Chapter 7. Using the MQSeries command sets

This chapter introduces the command sets that can be used to perform system administration tasks on MQSeries objects.

Administration tasks include creating, starting, altering, viewing, stopping, and deleting MQSeries objects such as queue managers, queues, processes, channels, and namelists. To perform these tasks, you must select the appropriate command from one of the supplied command sets.

Introducing command sets

MQSeries provides three command sets for performing administration tasks:
- Control commands
- MQSC commands
- PCF commands

This section describes the command sets that are available. Some tasks can be performed using either a control command or an MQSC command, but other tasks can be performed using only one type of command. For a comparison of the facilities provided by the different types of command set, see the *MQSeries System Administration* book.

Control commands

Control commands fall into three categories:

- *Queue manager commands*, including commands for creating, starting, stopping, and deleting queue managers and command servers.
- *Channel commands*, including commands for starting and ending channels and channel initiators.
- *Utility commands*, including commands associated with authority management and conversion exits.

Using control commands

In MQSeries in UNIX environments, you enter control commands in a shell window. In these environments, control commands, including the command name itself, the flags, and any arguments, are case sensitive. For example, in the command:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE jupiter.queue.manager
```

- The command name must be **crtmqm**, not **CRTMQM**.
- The flag must be `-u`, not `-U`.

- The dead-letter queue is `SYSTEM.DEAD.LETTER.QUEUE`.
- The argument is specified as `jupiter.queue.manager`, which is different from `JUPITER.queue.manager`.

Therefore, take care to type the commands exactly as you see them in the examples.

The following list contains a brief description of each of the control commands. You can obtain help for the syntax of any of the commands by entering the command followed by a question mark. MQSeries responds by listing the syntax required for the selected command.

crtmqcvx (data conversion)
Creates a fragment of code that performs data conversion on data type structures.

crtmqm (create queue manager)
Creates a local queue manager and defines the default and system objects.

dltmqm (delete queue manager)
Deletes a specified queue manager.

dmpmqlog (dump log)
Dumps a formatted version of the MQSeries system log.

dspmqaut (display authority)
Displays the current authorizations to a specified object.

dspmqcsv (display command server)
Displays the status of the command server for the specified queue manager.

dspmqfls (display MQSeries files)
Displays the real file system name for all MQSeries objects that match a specified criterion.

dspmqtrc (display MQSeries formatted trace output)
Displays MQSeries formatted trace output.

dspmqtrn (display MQSeries transactions)
Displays details of in-doubt transactions.

endmqcsv (end command server)
Stops the command server on the specified queue manager.

endmqlsr
Ends a listener process.

endmqm (end queue manager)
Stops a specified local queue manager.

endmqtrc (end MQSeries trace)
Ends tracing for the specified entity or all entities.

rcdmqimg (record media image)
Writes an image of an MQSeries object, or group of objects, to the log for use in media recovery.

rcrmqobj (recreate object)
: Recreates an object, or group of objects, from their images contained in the log.

rsvmqtrn (resolve MQSeries transactions)
: Commits or backs out internally or externally coordinated in-doubt transactions.

runmqchi (run channel initiator)
: Runs a channel initiator process.

runmqchl (run channel)
: Runs either a Sender (SDR) or a Requester (RQSTR) channel.

runmqdlq (run dead-letter queue handler)
: Starts the dead-letter queue (DLQ) handler, a utility that you can run to monitor and handle messages on a dead-letter queue.

runmqlsr (run listener)
: Runs a listener process.

runmqsc (run MQSeries commands)
: Issues MQSC commands to a queue manager.

runmqtrm (start trigger monitor)
: Invokes a trigger monitor.

setmqaut (set/reset authority)
: Changes the authorizations to an object or to a class of objects.

strmqcsv (start command server)
: Starts the command server for the specified queue manager.

strmqm (start queue manager)
: Starts a local queue manager.

strmqtrc (start MQSeries trace)
: Enables tracing.

For more information about the syntax and purpose of control commands, see the *MQSeries System Administration* book.

MQSeries (MQSC) commands

You use the MQSeries (MQSC) commands to manage queue manager objects, including the queue manager itself, channels, queues, and process definitions. For example, there are commands to define, alter, display, and delete a specified queue.

When you display a queue, using the DISPLAY QUEUE command, you display the queue *attributes*. For example, the MAXMSGL attribute specifies the maximum length of a message that can be put on the queue. The command does not show you the messages on the queue.

For detailed information about each MQSC command, see the *MQSeries MQSC Command Reference* book.

Running MQSC commands

You run MQSC commands by invoking the control command **runmqsc**. You can run MQSC commands:

- Interactively by typing them at the keyboard
- As a sequence of commands from a text file

For more information about using MQSC commands, see the *MQSeries System Administration* book.

PCF commands

MQSeries programmable command format (PCF) commands allow administration tasks to be programmed into an administration program. In this way you can create queues and process definitions, and change queue managers, from a program. PCF commands cover the same range of functions that are provided by the MQSC facility. You can therefore write a program to issue PCF commands to any queue manager in the network from a single node. In this way, you can both centralize and automate administration tasks.

Note: Unlike MQSC commands, PCF commands and their replies are not in a text format that you can read.

For a complete description of the PCF data structures and how to implement them, see the *MQSeries Programmable System Management* book.

Working with queue managers

This section describes how you can perform operations on queue managers, such as creating, starting, stopping, and deleting them. MQSeries provides control commands for performing these tasks.

Before you can do anything with messages and queues, you must create at least one queue manager.

Creating a default queue manager

The following command:

- Creates a default queue manager called saturn.queue.manager
- Creates the default and system objects automatically
- Specifies the names of both a default transmission queue and a dead-letter queue

```
crtmqm -q -d MY.DEFAULT.XMIT.QUEUE -u
SYSTEM.DEAD.LETTER.QUEUE  saturn.queue.manager
```

where:

-q Indicates that this queue manager is the default queue manager.

-d MY.DEFAULT.XMIT.QUEUE
Is the name of the default transmission queue.

-u SYSTEM.DEAD.LETTER.QUEUE
Is the name of the dead-letter queue.

saturn.queue.manager
Is the name of this queue manager. This must be the last parameter specified on the **crtmqm** command.

For more information about these attributes, see the *MQSeries System Administration* book.

Starting a queue manager

Although you have created a queue manager, it cannot process commands or MQI calls until it has been started. Start the queue manager by typing in this command:

```
strmqm saturn.queue.manager
```

The **strmqm** command does not return control until the queue manager has started and is ready to accept connect requests.

Stopping a queue manager

To stop a queue manager, use the **endmqm** command. For example, to stop a queue manager called `saturn.queue.manager` use this command:

```
endmqm saturn.queue.manager
```

Quiesced shutdown

By default, the above command performs a *quiesced shutdown* of the specified queue manager. This may take a while to complete—a quiesced shutdown waits until all connected applications have disconnected.

Use this type of shutdown to notify applications to stop; you are not told when they have stopped.

You can specify the `-w` flag if you require confirmation that the queue manager has stopped. For example:

```
endmqm -w saturn.queue.manager
```

The command prompt does not return until the queue manager has stopped.

Immediate shutdown

An *immediate shutdown* allows any current MQI calls to complete, but any new calls fail. This type of shutdown does not wait for applications to disconnect from the queue manager. Use this as the normal way to stop the queue manager, optionally after a quiesce period.

For an immediate shutdown, the command is:

```
endmqm -i saturn.queue.manager
```

Preemptive shutdown

Do not use this method unless all other attempts to stop the queue manager using the **endmqm** command have failed. This method can have unpredictable consequences for connected applications.

If an immediate shutdown does not work, you must resort to a *preemptive shutdown*, specifying the -p flag. For example:

```
endmqm -p saturn.queue.manager
```

This stops all queue manager code immediately.

Deleting a queue manager

To delete a queue manager called saturn.queue.manager, first stop it, then use the following command:

```
dltmqm saturn.queue.manager
```

Note: Deleting a queue manager is a serious step, because you also delete all resources associated with that queue manager, including all queues and their messages, and all object definitions.

Working with MQSeries objects

This section describes briefly how to use MQSC commands to create, display, change, copy, and delete MQSeries objects.

You can use the MQSC facility interactively (by entering commands at the keyboard) or you can redirect the standard input device (stdin) to run a sequence of commands from a text file. The format of the commands is the same in both cases. The examples included here assume that you will be using the interactive method.

For more information about using MQSC commands, see the *MQSeries System Administration* book. For a complete description of the MQSC commands, see the *MQSeries MQSC Command Reference* book.

Before you can run MQSC commands, you must have created and started the queue manager that is going to run the commands. For more information see "Creating a default queue manager" on page 50.

Using the MQSC facility interactively

To start using the MQSC facility interactively, use the **runmqsc** command. Open a shell and enter:

```
runmqsc
```

A queue manager name has not been specified; therefore the MQSC commands will be processed by the default queue manager. Now type in any MQSC commands, as required. For example:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

Continuation characters must be used to indicate that a command is continued on the following line:

- A minus sign (−) indicates that the command is to be continued from the start of the following line.
- A plus sign (+) indicates that the command is to be continued from the first nonblank character on the following line.

Command input terminates with the final character of a nonblank line that is not a continuation character. You can also terminate command input explicitly by entering a semicolon (;). (This is especially useful if you accidentally enter a continuation character at the end of the final line of command input.)

Feedback from MQSC commands

When you issue commands from the MQSC facility, the queue manager returns operator messages that confirm your actions or tell you about the errors you have made. For example:

```
AMQ8006: MQSeries queue created
  .
  .
  .
AMQ8405: Syntax error detected at or near end of command segment below:-
Z
```

The first message confirms that a queue has been created; the second indicates that you have made a syntax error.

These messages are sent to the standard output device. If you have not entered the command correctly, refer to the *MQSeries MQSC Command Reference* book for the correct syntax.

Ending interactive input to MQSC

To end interactive input of MQSC commands, enter the MQSC END command:

```
 END
```

Alternatively, you can use the EOF character `CTRL+D`.

If you are redirecting input from other sources, such as a text file, you do not have to do this.

Defining a local queue

For an application, the local queue manager is the queue manager to which the application is connected. Queues that are managed by the local queue manager are said to be local to that queue manager.

Use the MQSC command DEFINE QLOCAL to create a definition of a local queue and also to create the data structure that is called a queue. You can also modify the queue characteristics from those of the default local queue.

In this example, the queue we define, ORANGE.LOCAL.QUEUE, is specified to have these characteristics:

- It is enabled for gets, disabled for puts, and operates on a first-in-first-out (FIFO) basis.
- It is an 'ordinary' queue, that is, it is not an initiation queue or a transmission queue, and it does not generate trigger messages.
- The maximum queue depth is 1000 messages; the maximum message length is 2000 bytes.

The following MQSC command does this:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
       DESCR('Queue for messages from other systems') +
       PUT (DISABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (FIFO) +
       MAXDEPTH (1000) +
       MAXMSGL (2000) +
       USAGE (NORMAL);
```

Notes:

1. Most of these attributes are the defaults as supplied with the product. However, they are shown here for purposes of illustration. You can omit them if you are sure that the defaults are what you want or have not been changed. See also "Displaying default object attributes".
2. USAGE (NORMAL) indicates that this queue is not an initiation queue or a transmission queue.
3. If you already have a local queue on the same queue manager with the name ORANGE.LOCAL.QUEUE, this command fails. Use the REPLACE attribute if you want to overwrite the existing definition of a queue, but see also "Changing local queue attributes" on page 56.

Displaying default object attributes

When you define an MQSeries object, it takes any attributes that you do not specify from the default object. For example, when you define a local queue, the queue inherits any attributes that you omit in the definition from the default local queue, which is called SYSTEM.DEFAULT.LOCAL.QUEUE. The default local queue is created automatically when you create the default queue manager. To see exactly what these attributes are, use the following command:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

Note: The syntax of this command is different from that of the corresponding **DEFINE** command.

You can selectively display attributes by specifying them individually. For example:

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
        MAXDEPTH +
        MAXMSGL +
        CURDEPTH;
```

This command displays the three specified attributes as follows:

```
AMQ8409: Display Queue details.
    QUEUE(ORANGE.LOCAL.QUEUE)
    MAXDEPTH(1000)
    MAXMSGL(2000)
    CURDEPTH(0)
```

CURDEPTH is the current queue depth; that is, the number of messages on the queue. This is a useful attribute to display, because by monitoring the queue depth you can ensure that the queue does not become full.

Copying a local queue definition

You can copy a queue definition using the LIKE attribute on the **DEFINE** command.

For example:

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE)
```

This command creates a queue with the same attributes as our original queue ORANGE.LOCAL.QUEUE, rather than those of the system default local queue.

You can also use this form of the **DEFINE** command to copy a queue definition, but substituting one or more changes to the attributes of the original. For example:

```
DEFINE QLOCAL (THIRD.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE) +
       MAXMSGL(1024);
```

This command copies the attributes of the queue ORANGE.LOCAL.QUEUE to the queue THIRD.QUEUE, but specifies that the maximum message length on the new queue is to be 1024 bytes, rather than 2000.

Notes:

1. When you use the LIKE attribute on a **DEFINE** command, you are copying the queue attributes only. You are not copying the messages on the queue.
2. If you define a local queue, without specifying LIKE, it is the same as `DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE)`.

Changing local queue attributes

You can change queue attributes in two ways, using either the **ALTER QLOCAL** command or the **DEFINE QLOCAL** command with the REPLACE attribute. In “Defining a local queue” on page 54, we defined the queue ORANGE.LOCAL.QUEUE. Suppose, for example, you wanted to increase the maximum message length on this queue to 10 000 bytes.

- Using the **ALTER** command:

  ```
  ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
  ```

  This command changes a single attribute, that of the maximum message length; all the other attributes remain the same.
- Using the **DEFINE** command with the REPLACE option, for example:

  ```
  DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
  ```

  This command changes not only the maximum message length, but all the other attributes, which are given their default values. The queue is now put enabled whereas previously it was put inhibited. Put enabled is the default, as specified by the queue SYSTEM.DEFAULT.LOCAL.QUEUE, unless you have changed it.

  If you decrease the maximum message length on an existing queue, existing messages are not affected. Any new messages, however, must meet the new criteria.

Clearing a local queue

To delete all the messages from a local queue called MAGENTA.QUEUE, use the following command:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

You cannot clear a queue if:

- There are uncommitted messages that have been put on the queue under syncpoint.
- An application currently has the queue open.

Deleting a local queue

Use the MQSC command **DELETE QLOCAL** to delete a local queue. A queue cannot be deleted if it has uncommitted messages on it. However, if the queue has one or more committed messages, and no uncommitted messages, it can be deleted only if you specify the PURGE option. For example:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

Specifying NOPURGE instead of PURGE ensures that the queue is not deleted if it contains any committed messages.

Browsing queues

MQSeries provides a sample queue browser that you can use to look at the contents of the messages on a queue. The browser is supplied in both source and executable formats.

The default file names and paths are:

Source
/opt/mqm/samp/amqsbcg0.c

Executable
/opt/mqm/samp/bin/amqsbcg

The sample requires two input parameters, the queue manager name and the queue name. For example:

```
  amqsbcg ORANGE.LOCAL.QUEUE saturn.queue.manager
```

There are no defaults; both parameters are required.

Chapter 8. Using the MQSeries Internet Gateway

This chapter introduces the MQSeries Internet Gateway. It also explains how to get more information about using the MQSeries Internet Gateway.

The MQSeries Internet Gateway is one of the installable components on the MQSeries Server CD-ROM, and is also available from the MQSeries Web site.

The following Gateways are available:
- MQSeries Internet Gateway for AIX®
- MQSeries Internet Gateway for HP-UX
- MQSeries Internet Gateway for Linux
- MQSeries Internet Gateway for OS/2®
- MQSeries Internet Gateway for OS/390® OpenEdition®
- MQSeries Internet Gateway for Sun Solaris
- MQSeries Internet Gateway for Windows NT®

Overview of MQSeries Internet Gateway

MQSeries Internet Gateway provides a bridge between the synchronous World Wide Web and asynchronous MQSeries applications. With the MQSeries Internet Gateway, Web server software and MQSeries together provide an Internet-connected Web browser with access to MQSeries applications. This means that enterprises can take advantage of the low-cost access to global markets provided by the Internet, while benefitting from the robust infrastructure and assured message delivery of MQSeries.

User interaction with the MQSeries Internet Gateway is through HTML fill-out form POST requests; MQSeries applications respond by returning HTML pages to the MQSeries Internet Gateway, via an MQSeries queue.

The MQSeries Internet Gateway supports the following Web server interfaces:
- Common Gateway Interface (CGI)
- Internet Connection Application Programming Interface (ICAPI)
- Internet Services Application Programming Interface (ISAPI)
- Netscape Connection Application Programming Interface (NSAPI)

Note that:
- HP-UX does not support NSAPI.
- Sun Solaris does not support ISAPI.
- Linux supports CGI only.

MQSeries Internet Gateway documentation

The MQSeries product family Web site is at:

```
http://www.ibm.com/software/mqseries/
```

The following documentation is accessible from this Web site:

- *Getting Started with MQSeries Internet Gateway*. This is the starting point for the download and installation of MQSeries Internet Gateway.
- *MQSeries Internet Gateway User's Guide*. This is the main documentation for users of the MQSeries Internet Gateway.

Chapter 9. Obtaining additional information

This chapter describes the documentation for MQSeries for Linux. It starts with a list of the publications, and then discusses:

- "Hardcopy books"
- "Online information" on page 62

MQSeries for Linux is described in the following books:

Table 4. MQSeries for Linux books

Order Number	Title
	LINUX Specific Books
GC34-5691	*MQSeries for Linux Quick Beginnings*
	MQSeries Family Books
GC34-5761	*MQSeries V5.2 Release Guide*
SC33-1872	*MQSeries Intercommunication*
SC34-5349	*MQSeries Queue Manager Clusters*
GC33-1632	*MQSeries Clients*
SC33-1873	*MQSeries System Administration*
SC33-1369	*MQSeries MQSC Command Reference*
SC33-1482	*MQSeries Programmable System Management*
SC34-5390	*MQSeries Administration Interface Programming Guide and Reference*
GC33-1876	*MQSeries Messages*
SC33-0807	*MQSeries Application Programming Guide*
SC33-1673	*MQSeries Application Programming Reference*
SX33-6095	*MQSeries Programming Interfaces Reference Summary*
SC33-1877	*MQSeries Using C++*

Hardcopy books

The book that you are reading now is *MQSeries for Linux, V5.2 Quick Beginnings*. This book and the *MQSeries V5.2 Release Guide* are the only books that are supplied in hardcopy with the product. However, all books listed in Table 4 are available for you to order or print.

You can order publications from the IBMLink™ Web site at:

Hardcopy books

```
http://www.ibm.com/ibmlink
```

In the United States, you can also order publications by dialing **1-800-879-2755**.

In Canada, you can order publications by dialing **1-800-IBM-4YOU (1-800-426-4968).**

For further information about ordering publications contact your IBM authorized dealer or marketing representative.

For information about printing books, see “PDF” on page 63.

Online information

This section describes:
- “Publications supplied with the product”
- “HTML and PDF books on the World Wide Web” on page 64
- “BookManager CD-ROMs” on page 64
- “Online help” on page 64

Publications supplied with the product

On the product CD-ROM there is a directory called `books`. The books directory contains MQSeries books in HTML and PDF formats. To access them point your Web browser to `books/start.htm`.

HTML

You can view the MQSeries online documentation in HTML format directly from the CD-ROM. All books except for the *MQSeries Programming Interfaces Reference Summary* are available in U.S. English and also in some or all of the following national languages:
- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Spanish
- Simplified Chinese
- Traditional Chinese

When you read the books in HTML, you can follow hypertext links from one book to another. If you are reading translated books and link to a book that is not available in your national language, the U.S. English version of the book will be opened instead.

PDF

A PDF (Portable Document Format), corresponding to each hardcopy book, is available on the CD-ROM. You can read PDFs using Adobe Acrobat Reader. Also, you can download them to your own file system, or you can print them on a PostScript printer. If you have a Web browser, you can access the PDFs on the product CD-ROM by pointing your browser to `books/start.htm`.

The PDFs are available in U.S. English and also in some or all of the following national languages:
- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Spanish
- Simplified Chinese
- Traditional Chinese

To find out which ones are available in your language, look for the appropriate directory on the CD-ROM. The PDFs are in a subdirectory called ll_LL, where ll_LL is one of the following:
- pt_BR (Brazilian Portuguese)
- en_US (English)
- fr_FR (French)
- de_DE (German)
- it_IT (Italian)
- ja_JP (Japanese)
- ko_KR (Korean)
- es_ES (Spanish)
- zh_CN (Simplified Chinese)
- zh_TW (Traditional Chinese)

Within these directories, you can find the complete set of PDFs that are available. Table 5 shows the file names used for the PDF files.

Table 5. MQSeries publications – file names

Book	File Name
MQSeries V5.2 Release Guide	AMQZAY00
MQSeries Intercommunication	CSQZAE04
MQSeries Queue Manager Clusters	CSQZAH02
MQSeries Clients	CSQZAF04
MQSeries System Administration	AMQZAG01

Table 5. MQSeries publications – file names (continued)

Book	File Name
MQSeries MQSC Command Reference	CSQZAJ04
MQSeries Programmable System Management	CSQZAI03
MQSeries Administration Interface Programming Guide and Reference	CSQZAT01
MQSeries Messages	AMQZAO01
MQSeries Application Programming Guide	CSQZAL04
MQSeries Application Programming Reference	CSQZAK04
MQSeries Programming Interfaces Reference Summary	CSQZAM04
MQSeries Using C++	AMQZAN03

HTML and PDF books on the World Wide Web

The MQSeries books are available on the World Wide Web as well as on the product CD-ROM. They are available in PDF and HTML format. The MQSeries product family Web site is at:

```
http://www.ibm.com/software/mqseries/
```

By following links from this Web site you can:
- Obtain latest information about the MQSeries product family.
- Access the MQSeries books in HTML and PDF formats.
- Download MQSeries SupportPacs.

BookManager CD-ROMs

The MQSeries library is supplied in IBM BookManager® format on a variety of online library collection kits, including the *Transaction Processing and Data* collection kit, SK2T-0730. You can view the softcopy books in IBM BookManager format using the following IBM licensed programs:

BookManager READ/2
BookManager READ/6000
BookManager READ/DOS
BookManager READ/MVS
BookManager READ/VM
BookManager READ for Windows®

Online help

Man pages are provided for all API calls, MQSC commands, and relevant control commands including **crtmqm**, **strmqm**, and **endmqm**.

Part 3. Appendixes

Appendix A. Sample MQI programs and MQSC files

MQSeries for Linux provides a set of short sample MQI programs and MQSC command files. You can use these directly or modify them for experimental purposes.

MQSC command file samples

Table 6 lists the MQSC command file samples. These are simply ASCII text files containing MQSC commands. You can invoke the **runmqsc** command against each file in turn to create the objects specified in the file.

By default, these files are located in directory /opt/mqm/samp.

Table 6. MQSC Command Files

File name	Purpose
amqscos0.tst	Creates a set of MQI objects for use with the C program samples.
amqmdefs.tst	Defines objects for the administration application sample.

C Program Samples

Table 7 lists the sample MQI source files. By default, the source files are in /opt/mqm/samp and the compiled versions are in /opt/mqm/samp/bin. To find out more about what the programs do and how to use them, see the *MQSeries Application Programming Guide*.

Table 7. Sample Programs - Source Files

C source file	Purpose
amqsbcg0.c	Reads and then outputs both the message descriptor and message context fields of all the messages on a specified queue.
amqsecha.c	Echoes a message from a message queue to the reply-to queue. Can be run as a triggered application program.
amqsgbr0.c	Writes messages from a queue to `stdout`, leaving the messages on the queue. Uses MQGET with the browse option.
amqsget0.c	Removes the messages from the named queue (using MQGET) and writes them to `stdout`.
amqsinqa.c	Reads the triggered queue; each request read as a queue name; responds with information about that queue.

C samples

Table 7. Sample Programs - Source Files (continued)

C source file	Purpose
amqsput0.c	Copies `stdin` to a message and then puts this message on a specified queue.
amqsreq0.c	Puts request messages on a specified queue and then displays the reply messages.
amqsseta.c	Inhibits puts on a named queue and responds with a statement of the result. Runs as a triggered application.
amqstrg0.c	A trigger monitor that reads a named initiation queue and then starts the program associated with each trigger message. Provides a subset of the full triggering function of the supplied **runmqtrm** command.
amqsvfcx.c	A sample C skeleton of a Data Conversion exit routine.
amqsptl0.c	Putting messages to a distribution list.
amqsprma.c	Putting reference messages to a queue.
amqsgrma.c	Getting reference messages from a queue.
amqsxrma.c	Reference message channel exit.
Note: You can create the objects required by these samples using the MQSC command file amqscos0.tst.	

Supporting Databases

The database samples are located in the xatm subdirectory within the samples directory.

Table 8. Sample Programs - Databases

C source file	Purpose
amqsxas0.c	Updates a single database within an MQSeries unit of work.
amqsxag0.c	amqsxag0.c together with amqsxab0.sqc and amqsxaf0.sqc, or amqsxag0.cbl together with amqsxab0.sqb and amqsxaf0.sqb, update two databases within an MQSeries unit of work.

Miscellaneous Tools

| These tool files are provided to support the trace formatter and code conversion.

| *Table 9. Miscellaneous Files*

File name	Location	Purpose
amqtrc.fmt	/opt/mqm/lib	Defines MQSeries trace formats.

Table 9. Miscellaneous Files (continued)

File name	Location	Purpose
ccsid.tbl	/var/mqm/conv/table	Edit this file to add any newly supported CCSID values to your MQSeries system.

Appendix B. Code sets supported on MQSeries for Linux

MQSeries for Linux supports most of the code sets used by the locales – that is, the subsets of the user’s environment that define the conventions for a specific culture – that are provided as standard on Linux.

The CCSID (Coded Character Set Identifier) used in MQSeries to identify the code set used for the message and message header data is obtained by analyzing the code-set value returned by the locale.

If the locale is not set, CCSID 819 (the ISO 8859-1 code set) is used.

Table 10 shows the locales and the CCSIDs that are registered for the code set used by the locale. Note that not all the locales listed below are supported by all versions of Linux.

Table 10. Locales and CCSIDs

Locale	**Language**	**Code Set**	**CCSID**
C	English	ISO 8859-1	819
POSIX	English	ISO 8859-1	819
cs	Czech	ISO 8859-2	912
cs_CZ	Czech	ISO 8859-2	912
czech	Czech	ISO 8859-2	912
da	Danish	ISO 8859-1	819
da_DK	Danish	ISO 8859-1	819
danish	Danish	ISO 8859-1	819
de	German	ISO 8859-1	819
de.ISO8859-15	German	ISO 8859-15	923
de_DE	German	ISO 8859-1	819
german	German	ISO 8859-1	819
de_AT	German - Austria	ISO 8859-1	819
de_AT.ISO8859-15	German - Austria	ISO 8859-15	923
de_BE	German - Belgium	ISO 8859-1	819
de_CH	German - Switzerland	ISO 8859-1	819
de_LU	German - Luxembourg	ISO 8859-1	819
el	Greek	ISO 8859-7	813
el_GR	Greek	ISO 8859-7	813
greek	Greek	ISO 8859-7	813
en	English - United Kingdom	ISO 8859-1	819
en_GB	English - United Kingdom	ISO 8859-1	819

Table 10. Locales and CCSIDs (continued)

Locale	Language	Code Set	CCSID
en_AU	English - Australia	ISO 8859-1	819
ca	English - Canada	ISO 8859-1	819
en_CA	English - Canada	ISO 8859-1	819
en_DK	English - Denmark	ISO 8859-1	819
en_IE	English - Ireland	ISO 8859-1	819
en_US	English - USA	ISO 8859-1	819
es	Spanish	ISO 8859-1	819
es.ISO8859-15	Spanish	ISO 8859-15	923
es_ES	Spanish	ISO 8859-1	819
spanish	Spanish	ISO 8859-1	819
es_DO	Spanish - Dominican Republic	ISO 8859-1	819
es_GT	Spanish - Guatemala	ISO 8859-1	819
es_HN	Spanish - Honduras	ISO 8859-1	819
es_MX	Spanish - Mexico	ISO 8859-1	819
es_PA	Spanish - Panama	ISO 8859-1	819
es_PE	Spanish - Peru	ISO 8859-1	819
es_SV	Spanish - El Salvador	ISO 8859-1	819
et	Estonian	ISO 8859-1	819
et_EE	Estonian	ISO 8859-1	819
eu_ES	Basque - Spain	ISO 8859-1	819
fi	Finnish	ISO 8859-1	819
fi_FI	Finnish	ISO 8859-1	819
finnish	Finnish	ISO 8859-1	819
fo_FO	Faeroese	ISO 8859-1	819
fr	French - France	ISO 8859-1	819
fr.ISO8859-15	French - France	ISO 8859-15	923
fr_FR	French - France	ISO 8859-1	819
french	French - France	ISO 8859-1	819
fr_BE	French - Belgium	ISO 8859-1	819
fr_BE.ISO8859-15	French - Belgium	ISO 8859-15	923
fr_CA	French - Canada	ISO 8859-1	819
fr_CH	French - Switzerland	ISO 8859-1	819
fr_LU	French - Luxembourg	ISO 8859-1	819
ga	Gaelic	ISO 8859-1	819
ga_IE	Gaelic	ISO 8859-1	819

Table 10. Locales and CCSIDs (continued)

Locale	Language	Code Set	CCSID
hr	Croatian	ISO 8859-2	912
hr_HR	Croatian	ISO 8859-2	912
hu	Hungarian	ISO 8859-2	912
hu_HU	Hungarian	ISO 8859-2	912
hungarian	Hungarian	ISO 8859-2	912
in	Indonesian	ISO 8859-1	819
in_ID	Indonesian	ISO 8859-1	819
indonesian	Indonesian	ISO 8859-1	819
is	Icelandic	ISO 8859-1	819
is_IS	Icelandic	ISO 8859-1	819
icelandic	Icelandic	ISO 8859-1	819
it	Italian - Italy	ISO 8859-1	819
it.ISO8859-15	Italian - Italy	ISO 8859-15	923
it_IT	Italian - Italy	ISO 8859-1	819
italian	Italian - Italy	ISO 8859-1	819
it_CH	Italian - Switzerland	ISO 8859-1	819
iw	Hebrew	ISO 8859-8	916
iw_IL	Hebrew	ISO 8859-8	916
hebrew	Hebrew	ISO 8859-8	916
ja	Japanese	eucJP	5050
ja_JP	Japanese	eucJP	5050
ja_JP.EUC	Japanese	eucJP	5050
ja_JP.PCK	Japanese	PCK	943
kl_GL	Greenlandic	ISO 8859-1	819
ko	Korean	eucKR	970
lt_LT	Lithuanian	ISO 8859-1	819
lv_LV	Lettish	ISO 8859-1	819
nl	Dutch - Netherlands	ISO 8859-1	819
nl_NL	Dutch - Netherlands	ISO 8859-1	819
dutch	Dutch - Netherlands	ISO 8859-1	819
nl_BE	Dutch - Belgium	ISO 8859-1	819
no	Norwegian	ISO 8859-1	819
no_NO	Norwegian	ISO 8859-1	819
norwegian	Norwegian	ISO 8859-1	819
pl	Polish	ISO 8859-2	912
pl_PL	Polish	ISO 8859-2	912
polish	Polish	ISO 8859-2	912

Table 10. Locales and CCSIDs (continued)

Locale	Language	Code Set	CCSID
pt pt_PT portuguese	Portuguese Portuguese Portuguese	ISO 8859-1 ISO 8859-1 ISO 8859-1	819 819 819
pt_BR	Portuguese - Brazil	ISO 8859-1	819
ro ro_RO romanian	Romanian Romanian Romanian	ISO 8859-2 ISO 8859-2 ISO 8859-2	912 912 912
ru ru_RU russian	Russian Russian Russian	ISO 8859-5 ISO 8859-5 ISO 8859-5	915 915 915
ru_SU	Russian - (former) USSR	ISO 8859-5	915
ru_UA	Russian - Ukraine	ISO 8859-5	915
sk sk_SK slovak	Slovak Slovak Slovak	ISO 8859-2 ISO 8859-2 ISO 8859-2	912 912 912
sl sl_SI slovene	Slovene Slovene Slovene	ISO 8859-2 ISO 8859-2 ISO 8859-2	912 912 912
sr sk_YU	Serbian Serbian	ISO 8859-5 ISO 8859-5	915 915
sv sv_SE swedish	Swedish Swedish Swedish	ISO 8859-1 ISO 8859-1 ISO 8859-1	819 819 819
sv_FI	Swedish - Finland	ISO 8859-1	819
tr tr_TR turkish	Turkish Turkish Turkish	ISO 8859-9 ISO 8859-9 ISO 8859-9	920 920 920
uk uk_UA	Ukrainian Ukrainian	ISO 8859-5 ISO 8859-5	915 915
zh	Simplified Chinese	eucCN	1383
zh_TW zh_TW.BIG5	Traditional Chinese Traditional Chinese	eucTW BIG5	964 950

For further information about interplatform support for these locales, see the *MQSeries Application Programming Reference* book. Data-conversion information in that book applicable to MQSeries for Sun Solaris V5.2 applies also to MQSeries for Linux, V5.2.

Appendix C. Building applications on Linux

This appendix provides information specific to MQSeries for Linux about:

- Preparing C programs
- Preparing C++ programs
- Linking C and C++ libraries
- Building user exits

Preparing C programs

The MQSeries C include files are listed in Table 11. They are installed in the directory /opt/mqm/inc/. The include files are symbolically linked into /usr/include.

Table 11. C Include Files for MQSeries

File name	Contents
<cmqc.h>	Call prototypes, data types, structures, return codes, and constants
<cmqcfc.h>	Definitions for programmable commands (PCFs)
<cmqxc.h>	Definitions for channel exits and data-conversion exits
<cmqzc.h>	Definitions for installable services
Note: The files are protected against multiple declaration, so you can include them many times.	

Compiling C programs

Precompiled C programs are supplied in the /opt/mqm/samp/bin directory. To build a sample from source code, use the gcc compiler.

For example, to compile the sample program amqsput0:

```
gcc -o <amqsput0> <amqsput0>.c -lmqm
```

If you want to use the programs on a machine that has only the MQSeries for Linux client installed, recompile the programs to link them with the client library instead. That is, for single-threaded applications:

```
gcc -o <amqsput0> <amqsput0>.c -lmqic
```

Preparing C++ programs

For information about using C++, see the *MQSeries Using C++* book. Please use the following information in conjunction with the information in that book.

Compilers for MQSeries for Linux

Platform	Compiler	Switches	Libraries
Linux	g++ (GNU)		-lmqb23gl -lmq{c|s}23gl libmqb23gl_r.so libmqs23gl_r.so libmqb23gl_r.so libmqc23gl_r.so

Linking libraries

You must link with the MQSeries libraries that are appropriate for your application type: if you want to run multithreaded applications, you must link with a multithreaded version of the MQSeries libraries.

Program type	Single-threaded library files	Multithreaded library files
Server for C	libmqm.so	libmqm_r.so
Client for C	libmqic.so	libmqic_r.so
Server for C++	libmqb23gl.so, libmqs23gl.so	libmqb23gl_r.so, libmqs23gl_r.so
Client for C++	libmqb23gl.so, libmqc23gl.so	libmqb23gl_r.so, libmqc23gl_r.so

Note: If you are writing an installable service (as described in the *MQSeries Programmable System Management* book) link to the libmqmzf_r.so library.

Selecting the libraries to match your compiler

When you are using the GNU C++ compiler you must make sure that any libraries containing C++ functions are built using the same version of the C++ complier as your application. To give you as much flexibility as possible, and to adhere to this condition, MQSeries contains three versions of the MQSeries C++ interface libraries. Each one matches a supported level of the GNU compiler.

The supported versions of the compiler are:

- 2.7.2.3
- egcs-2.91.66 (the default)
- 2.952

Libraries that match these versions are in the /opt/mqm/lib/<version> directory. Links are created from the default version to the /opt/mqm/lib directory.

To verify the version of the compiler that you are using, enter the command

```
g++ --version
```

If your chosen compiler version does not match the default version selected by MQSeries, you must reconfigure the compiler to use the correct libraries by linking the C++ libraries in the MQSeries lib directory to those that match your compiler. To do this, issue the following commands replacing <version> with the version of your compiler.

```
ln -s -f /opt/mqm/lib/<version>/libimqb23gl.so /opt/mqm/lib/libimqb23gl.so
ln -s -f /opt/mqm/lib/<version>/libimqs23gl.so /opt/mqm/lib/libimqs23gl.so
ln -s -f /opt/mqm/lib/<version>/libimqc23gl.so /opt/mqm/lib/libimqc23gl.so
ln -s -f /opt/mqm/lib/<version>/libimqb23gl_r.so /opt/mqm/lib/libimqb23gl_r.so
ln -s -f /opt/mqm/lib/<version>/libimqs23gl_r.so /opt/mqm/lib/libimqs23gl_r.so
ln -s -f /opt/mqm/lib/<version>/libimqc23gl_r.so /opt/mqm/lib/libimqc23gl_r.so
```

Appendix D. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make

Trademarks

Index

A

administration command sets
- control commands 47
- MQSeries commands (MQSC) 49
- programmable command format commands (PCF) 50

AMQ_INHIBIT_DLCLOSE 16
amqsget sample program 18
amqsgetc sample program 30
amqsput sample program 18
amqsputc sample program 30
attributes
- ALL attribute 54
- changing 56
- default 54

B

bibliography 61
BookManager 64
books
- ordering 61
- printing 63

books, translated 15
browsing queues 57
building applications 75

C

C++ programs 76
C language include files 75
C sample programs 67
Caldera OpenLinux 4
capabilities of MQSeries 44
case-sensitive control commands 47
CCSID (coded character set identifier)
- setting 24
- supported on MQSeries for Linux 71

changing queue attributes 56
channel
- events 46
- message 42
- MQI 20, 28, 42
- receiver 22
- sender 20

clearing a local queue 56
client channel 42
client-connection channel, example 29
client installation 25
client-server configurations 43
client setup, example 29
clients 43
clusters 43
code sets, supported by MQSeries for Linux 71
coded character set identifier (CCSID)
- setting 24
- supported on MQSeries for Linux 71

command set administration 47
commands
- control 47
- MQSC
  - ALTER QLOCAL 56
  - DEFINE QLOCAL 55
  - DEFINE QLOCAL LIKE 55
  - DEFINE QLOCAL REPLACE 56
  - DELETE QLOCAL 57
  - using 50
- programmable command format (PCF) 50
- runmqsc 53

compilers 4, 25
compiling C programs 75
compiling user exits 23
configuration, kernel 13
configurations 41
control commands
- case-sensitive 47
- runmqsc 53

controlled shutdown 51
creating
- file system for product code 13
- groups
  - client 26
  - server 11
- queue manager 50
- users 11, 26

current queue depth (CURDEPTH) 55

D

DB2 Universal Database 5
default
- attributes of objects 54

default *(continued)*
- queue manager commands processed 53

deleting
- local queue 57
- MQSeries for Linux client 31
- queue manager 30, 52

disk requirements for installation
- client 25
- server 3

dltmqm command 30

E

ending
- interactive MQSC commands 53
- queue manager 51

endmqm command 30, 51
environment variable
- AMQ_INHIBIT_DLCLOSE 16
- LANG 16, 27
- MQSERVER 29
- NLSPATH 16, 27

error messages 53
events 45
- channel 46

example
- client-connection channel, defining 29
- client setup 29
- getting the message from the queue
  - on the MQSeries client 30
- inetd setup 20, 21, 28
- local queue, creating 22, 28
- MQSC, starting 19, 28
- MQSC, stopping 20, 28
- putting a message on the queue
  - on the MQSeries client 30
- queue manager
  - creating 28
  - starting 28
- server-connection channel, creating 28
- setting up the server 27
- verification, ending 30

F

feedback from MQSC commands 53

file samples
  miscellaneous 68
  MQSC 67
file system, creating for product code 13

G

groups, creating 11, 26

H

hard disk requirements 3
hardware requirements
  MQSeries for Linux client 25
  MQSeries for Linux server 3
HTML books 62
Hypertext Markup Language (HTML) 64

I

inetd setup 20, 21, 28
information, ordering publications 61
inodes, maximum 12, 13
installation
  client 26
  directory, client 26
  directory, server 11
  kernel configuration 13
  preparation 11, 26
  procedure 15, 26
  server 11
installing
  clients on the server 16
instrumentation events 45
interactive MQSC
  ending 53
  feedback from 53
  using 52
Internet Gateway 59
introduction to MQSeries 39

J

Java support, for MQSeries 15

K

kernel configuration 13

L

LANG environment variable 16, 27
LIKE attribute 55
linking libraries 76
Linux distributions 4
local queue 41
local queue manager 41
local queues
  clearing 56
  copying definitions 55
local queues *(continued)*
  defining one 54
  deleting 57
locales 71

M

maintenance 33
maximum open files and inodes 13
maximum processes 14
message
  channels 42
  description 40
  descriptor 40
message-driven processing 46
message length, decreasing 56
messages, translated 16, 27
monitoring queue managers 45
MQAI (MQSeries administration interface) 46
MQI channel 42
MQSC commands
  ALTER QLOCAL 56
  DEFINE QLOCAL 55
  DEFINE QLOCAL LIKE 55
  DEFINE QLOCAL REPLACE 56
  DELETE QLOCAL 57
  ending interactive input 53
  issuing interactively 52
  using 50
MQSeries for Linux
  applying maintenance 33
  client hardware requirements 25
  client installation 25
  client software requirements 25
  compilers supported 4
  components 5
    client 26
    server 15
  databases supported 5
  open files and inodes, maximum 13
  overview of 3
  processes, maximum 14
  README file 9
  restoring previous service level 33
  server hardware requirements 3
  server installation 11
  server software requirements 4
  TCP support 4
  verifying server installation 17
  web servers supported 5
MQSeries Web site 60
MQSERVER environment variable 29

N

national language support 16, 27
NLSPATH environment variable 16, 27
NR_TASKS, altering 14

O

objects
  default attributes 54
  working with 52
online books 62
online help 64
open files and inodes, maximum 13
ordering books 61
ordering publications 61
overview of MQSeries for Linux 3

P

PDF (Portable Document Format) 63
performance events 46
Portable Document Format (PDF) 63
preemptive queue manager shutdown 52
printing books 63
processes, maximum 14
program samples 67
  C 67
  databases 68
programmable command format (PCF)
  administration with 50
programming with MQSeries 46
publications 61

Q

queue depth
  current 55
  determining 55
queue manager
  creating 50
  definition 28
  deleting 30, 52
  description 41
  events 46
  immediate shutdown 51
  monitoring 45
  preemptive shutdown 51, 52
  shutdown
    controlled 51
    immediate 51
    preemptive 51
    quiesced 51
  starting 28, 51
  stopping 30, 51

queues
- attributes 40
- browsing 57
- changing attributes 56
- description 40
- local
  - clearing 56
  - copying 55
  - defining 54
  - deleting 57

quiesced shutdown 51

R

README File 9
Red Hat Linux 4
remote queue 41
remote queue manager 41
requirements, hardware and software 3
RPM images
- MQSeries for Linux client 8
- MQSeries for Linux server product 6

runmqsc
- ending 53
- feedback 53
- using interactively 52

S

S.u.S.E. Linux 4
sample files
- miscellaneous 68
- MQSC 67

sample programs 67
- C 67
- databases 68

server-client configurations 43
server-connection channel, example 28
server installation 11
setting the CCSID (coded character set identifier) 24
setting up the server, example 27
shell commands for MQSeries 47
shutdown queue manager
- controlled 51
- immediate 51
- preemptive 51
- quiesced 51

softcopy information 62
software requirements
- MQSeries for Linux client 25
- MQSeries for Linux server 4

space requirements, server installation 3
starting a queue manager 51
stopping a queue manager 30, 51
strmqm command 51
support, Java 15
supported code sets 71
syntax error, in MQSC commands 53

T

TCP support 4
transactional support 44
translated books 15
translated messages
- client 27
- server 16

triggering 46
TurboLinux 4

U

user exits, compiling 23
users, creating 11, 26

V

verification, ending 30
verify installation 27
verifying server installation 17

W

web servers 5
World Wide Web interface 59

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom

- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC34-5691-01

Spine information:

IBM MQSeries® for Linux MQSeries for Linux V5.2 Quick Beginnings Version 5.2