MQSeries® for Sun Solaris, Intel® Platform Edition

Quick Beginnings

Version 5 Release 1

GC34-5851-00

MQSeries® for Sun Solaris, Intel® Platform Edition

Quick Beginnings

Version 5 Release 1

GC34-5851-00

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix G. Notices" on page 83.

First edition (October 2000)

This edition applies to MQSeries for Sun Solaris, Intel Platform Edition, Version 5.1 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

Welcome to MQSeries for Sun Solaris, Intel Platform Edition

This book describes MQSeries for Sun Solaris, Intel Platform Edition, and explains how to plan for, install, and use the product.

In the rest of this book, the product is referred to as "MQSeries for Sun Solaris, Intel".

Road Map

Use Table 1 to find the information you need to get started with *MQSeries for Sun Solaris, Intel*.

Table 1. Getting Started Road Map

If you want to...	Refer to...
Read about MQSeries for Sun Solaris, Intel	"Chapter 1. About MQSeries for Sun Solaris, Intel" on page 1
Learn about system requirements for MQSeries for Sun Solaris, Intel	"Chapter 2. Planning to Install the MQSeries for Sun Solaris, Intel Server" on page 9
Install MQSeries for Sun Solaris, Intel	"Chapter 3. Installing the MQSeries for Sun Solaris, Intel Server" on page 15
Learn about system requirements for clients	"Chapter 4. Planning to Install MQSeries Clients" on page 29
Install an MQSeries client	"Chapter 5. Installing the MQSeries for Sun Solaris, Intel Client" on page 31
Start using command sets	"Chapter 6. Using the MQSeries Command Sets" on page 35
Start using the Web Interface	"Chapter 7. Using the MQSeries World Wide Web Interface" on page 47
View or print online documentation	"Chapter 8. Obtaining Additional Information" on page 49
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.
- `Monospace type` indicates an example (such as a fictitious path or file name) or text that is displayed on the screen.

Chapter 1. About MQSeries for Sun Solaris, Intel

This chapter introduces IBM MQSeries and describes its relationship with other products. It contains basic explanations of the following topics:

For more detailed explanations of these topics see the *MQSeries Planning Guide*.

Message Queuing

MQSeries enables applications to use message queuing to participate in message-driven processing. Applications can communicate across different platforms by using the appropriate message queuing software products. The applications are shielded from the mechanics of the underlying communications.

MQI – a Common Application Programming Interface

All MQSeries products implement a common application programming interface (message queue interface or MQI), regardless of the platform on which the applications are run. The calls made by the applications and the messages they exchange are common. This makes it much easier to write and maintain applications than it is when using traditional methods. It also makes it easier to port applications from one platform to another.

The MQI is described in detail in the *MQSeries Application Programming Reference* book.

Time-Independent Applications

With message queuing, the exchange of messages between the sending and receiving programs is time independent. This means that the sending and receiving applications are decoupled so that the sender can continue processing without having to wait for the receiver to acknowledge the receipt of the message. In fact, the target application does not even have to be running when the message is sent. It can retrieve the message after it is started.

Message Queuing

Message-Driven Processing

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

Messages and Queues

Messages and queues are the basic components of a message queuing system.

What is a Message?

A *message* is a string of bytes that has meaning to the applications that use it. Messages are used for transferring information 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.

What is a Queue?

A *queue* is a data structure that stores messages. The messages may be put on 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 and receive messages using MQI calls. For example, one application can put a message on a queue using the MQPUT call, and another application can retrieve the message from the same queue using the MQGET call.

MQSeries Objects

An MQSeries object is a recoverable resource managed by MQSeries. Many of the tasks described in this chapter involve manipulating the following types of MQSeries object:

- Queue managers
- Queues
- Process definitions
- Channels
- Namelists

For system administrators, commands are available to manipulate objects. Default objects are created for you when you create a queue manager.

Each object has a *name* associated with it and can be referenced in MQSeries commands and MQI calls by that name. Names must be unique within each of the object types. For example, you can have a queue and a process with the same name, but you cannot have two queues with the same name.

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 commands received.
- Special events such as trigger events or instrumentation events are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application making the MQPUT call. 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 simply 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.

Queues

A queue is an MQSeries object that can store messages. Each queue has *queue attributes* that determine what happens when applications reference the queue in MQI calls.

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)

Process Definitions

A *process definition object* defines an application that is to be started in response to a *trigger event* on an MQSeries queue manager.

A trigger event is a logical combination of conditions that is detected by a queue manager. For example, a trigger event may be generated when the number of messages on a queue reaches a predefined level. This event causes the queue manager to put a trigger message on a specified initiation queue. This trigger message is retrieved by a *trigger monitor*, a special application that monitors an initiation queue. The trigger monitor then starts up the application program that was specified in the trigger message.

If a queue manager is to use triggering, at least one initiation queue must be defined for that queue manager.

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

Channels

A channel provides a communication path. 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 the transmission of 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. If two-way communication is required between two queue managers, two message channels are required.

An *MQI channel* connects an MQSeries client to a queue manager on a server machine. It is for the transfer of MQI calls (for example, MQPUT) and

responses only and is bidirectional. A channel definition exists for each end of the link. On some platforms, some types of MQI channel can be defined automatically.

For more information on channels and how to use them, see the *MQSeries Intercommunication* book.

Namelists

A *namelist* is an MQSeries object that contains a list of other MQSeries objects. Typically, namelists are used by applications such as trigger monitors, where they are used to identify a group of queues. The advantage of using a namelist is that it is maintained independently of applications; that is, it can be updated without stopping any of the applications that use it. Also, if one application fails, the namelist is not affected and other applications can continue using it.

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 MQI calls from applications and pass them to an *MQI 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 *MQI 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 MQI server machine. A server can support local MQSeries applications as well.

The difference between an MQI server and an ordinary queue manager is that the MQI server can support MQI clients, and each MQI client has a dedicated communications link with the MQI server. For more information about creating channels for clients and servers or information about client support in general, see the *MQSeries Clients* book.

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, the following conditions give rise to a *Queue Full* event:

- Queue Full events are enabled for a specified queue, and
- An application issues an MQPUT call to put a message on that queue, but 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
- An application attempting to open a queue and specifying a user ID that is not authorized

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.

Types of Event

MQSeries events are categorized as follows:

Queue manager events
: These events 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 events 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 events 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.

Transactional Support

An application program can group a set of updates into a *unit of work*. These updates are usually logically related and must all be successful for data integrity to be preserved. If one update succeeded while another failed then data integrity would be lost.

A unit of work *commits* when it completes successfully. At this point all updates made within that unit of work are made permanent or irreversible. If the unit of work fails then all updates are instead *backed out*. *Syncpoint coordination* is the process by which units of work are 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 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 MQI 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 over queue manager restart.

Chapter 2. Planning to Install the MQSeries for Sun Solaris, Intel Server

This chapter is a summary of the requirements for running MQSeries for Sun Solaris, Intel, the network protocols and the compilers supported, the delivery media, and the various components of the product.

For current information on known problems and available fixes please see the MQSeries Family support page at:

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

Year 2000 compatibility

MQSeries, when used in accordance with its associated documentation, is capable of correctly processing, providing, and/or receiving date data within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software, and firmware) used with this IBM Program properly exchange accurate date data with it.

Customers should contact third party owners or vendors regarding the readiness status of their products.

IBM reserves the right to update the information shown here. For the latest information regarding levels of supported software, refer to:

`http://www.ibm.com/software/mqseries/platforms/supported.html`

For the latest IBM statement regarding Year 2000 readiness, refer to:

`http://www.ibm.com/IBM/year2000/`

Hardware Requirements

For details of the appropriate Intel hardware on which you can install the MQSeries for Sun Solaris, Intel, server and client, go to the Compatibility List section from the following internet page:

`http://access1.sun.com`

Then refer to the *Solaris 7 Intel Platform Edition Hardware Compatibility List*.

Hardware Requirements

- MQSeries Servers:

  Any Intel-based PC server or desktop system that is compatible with Sun Solaris 7 Intel Platform Edition.

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 or not. You also require archiving capacity on disk, tape, or other media.

These are the approximate storage requirements:

- Server:

  A minimum of 25 MB of disk space must be available for the product code and data in the filesystem containing the `/opt` directory.

  In addition, if you install the online books in HTML format you require 35 MB of storage for the books in the `/opt` directory.

  After installation the books are placed in the `/opt/mqm/html` directory.

Working data for MQSeries for Sun Solaris, Intel, 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

IBM MQSeries for Sun Solaris, Intel, Version 5.1 runs on:

- Sun Solaris 7 Intel Platform Edition Hardware 11/99 with patches 106981-10, 106542-10, 107637-03, and 107172-05

Note: No DCE or TXSeries™ products are supported.

Connectivity

The TCP/IP network protocol is supported.

- Any communications hardware supporting TCP/IP

TCP/IP is part of the base operating system.

Compilers Supported for Solaris, Intel Applications

- Sun WorkShop Compiler C Version 5.0 (in the Sun Visual Workshop Version 5.0, with patches 107296-01, 107830-02, 107290-03, 107361-04)
- Sun WorkShop Compiler C++ Version 5.0 (with patches 106328-08, 107312-09)
- Micro Focus COBOL compiler Version 4.1.20 for UNIX®

Options

You may use the following options with MQSeries for Sun Solaris, Intel.

Transaction Monitors

The following transaction processing monitor (coordination can be through X/Open XA interface) may be used:

- BEA Tuxedo Version 6.5

Databases

- Oracle 8i (Version 8.1.5)
- Oracle Version 8.0.6

Delivery

MQSeries for Sun Solaris, Intel, is supplied on CD-ROM.

Installation

MQSeries for Sun Solaris, Intel takes approximately 5 minutes to install, using the Solaris **pkgadd** program.

MQSeries for Sun Solaris, Intel Components

When you install MQSeries for Sun Solaris, Intel, you can choose which components to install. The components are as follows:

MQSeries Server
The MQSeries server provides support for client connections. It also provides support to enable you to create and support your own applications and external applications.

MQSeries Clients

MQSeries client for Sun Solaris, Intel and MQSeries client for Java™:

- The MQSeries client for Sun Solaris, Intel can be installed on the server machine, enabling you to have the MQSeries server and client on the same machine.
- The MQSeries client for Java allows Java applets running on your machine to communicate with MQSeries. It includes security exits for encryption and authentication of messages sent across the web by the MQSeries client for Java. These exits consist of some Java classes.

Notes:

1. When you have installed and configured your machine to use this version of the MQSeries client for Java, you may not be able to run applets, that were written using an earlier version of the MQSeries client for Java, from a browser.

   This is because the browser picks up the locally installed version of the MQSeries client for Java class files from your CLASSPATH statement, and these files are incompatible with earlier releases.

   If you want to run your old applets, remove the MQSeries client for Java library from your CLASSPATH statement and restart your web browser.
2. The Java client requires a Java 1.1.1 (or later) capable Web Browser. This may not be available on all platforms.

   Some providers of Java place limitations on its support or warranty. Read the documentation from the Java provider to understand any limitations on its use.

MQSeries Online Documentation

Online versions of the books for MQSeries for Sun Solaris, Intel in HTML format. Included are:

- MQSeries Documentation
- MQSeries Internet Documentation

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

HTML and PDF 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.

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

Samples
Sample application programs.

MQSeries code
The MQSeries code in the national languages listed under the **MQSeries Online Documentation** component.

Note: The "base" product is automatically installed.

Creating the System Default Objects

When you use the **crtmqm** command to create a queue manager with this release of MQSeries, the system default objects are automatically created. The sample MQSC definition file, amqscoma.tst, is no longer provided.

README file

Before starting to install MQSeries for Sun Solaris Intel, review the README file, which you will find in the root directory of the CD-ROM.

Migrating from an Earlier Version of MQSeries for Sun Solaris, Intel

If you want to migrate from MQSeries for Sun Solaris, Intel, V5.0 to MQSeries for Sun Solaris, Intel, V5.1, you should follow this procedure:

1. End all queue manager activity. Do this with the **endmqm** command. See "Stopping a Queue Manager" on page 39 for information on how to use the **endmqm** command.
2. Stop all MQSeries activity and remove any shared resources that MQSeries uses. Do this by shutting down the system and restarting it, or use the **ipcs -a** command to display shared memory segments or semaphore sets created by MQSeries, and remove them using the **ipcrm** command. You do not have to recreate your MQSeries objects.
3. Uninstall the old MQSeries for Sun Solaris, Intel using the **pkgrm** program. Do not delete the /var/mqm directory tree if you want to retain your own MQSeries information, for example your queue manager data.

You are now ready to install MQSeries for Sun Solaris, Intel. See "Installing the MQSeries for Sun Solaris, Intel Server" on page 17 for information on installation.

Chapter 3. Installing the MQSeries for Sun Solaris, Intel Server

This chapter tells you how to install MQSeries for Sun Solaris, Intel 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 16.

Note: The MQSeries product is contained in the ***/mq_solaris*** directory of the CD-ROM.

Preparing for Installation

This section guides you through some of the steps you must perform before you install MQSeries for Sun Solaris, Intel.

Before Installation

Before you can install MQSeries for Sun Solaris, Intel you:

- Must install any patches listed in the README or on the following Web sites:

  `http://www.ibm.com/software/mqseries/support/`
  `http://www.ibm.com/software/mqseries/platforms/supported.html`
- Must create a group with the name `mqm`
- Must add **root** to the `mqm` group
- Must create a user ID with the name `mqm`
- Are recommended to create and mount a `/var/mqm` file system, or `/var/mqm`, `/var/mqm/log`, and `/var/mqm/errors` file systems

  You should allow a minimum of 30 MB of storage for `/var/mqm`, 2 MB of storage for `/var/mqm/errors`, and 20 MB of storage for `/var/mqm/log` if you are creating separate file systems.

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

Notes:

1. The size of 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.

After installation, this 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, if that file system is 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.

Installing the MQSeries for Sun Solaris, Intel Server

This section describes the installation of the MQSeries for Sun Solaris, Intel server.

Notes:

1. If you have previously installed MQSeries on your system, you need to remove the product using the **pkgrm** program. See "Migrating from an Earlier Version of MQSeries for Sun Solaris, Intel" on page 14 for more information.
2. If the product is present, but not installed correctly, you may need to manually delete the files and directories contained in:

   ```
   /var/mqm
   /opt/mqm
   ```

Carry out the following procedure:

1. Check to see if Volume Manager is running on your system by typing the following command:

   ```
   /usr/bin/ps -ef | /bin/grep vold
   ```

   If it is running, the CD is mounted on /cdrom/mq_solaris automatically. If it is not running, mount the CD by typing the following commands:

   ```
   mkdir -p /cdrom/mq_solaris
   mount -F hsfs -r /dev/dsk/cntndnsn /cdrom/mq_solaris
   ```

   substituting `cntndnsn` with the name of your CD-ROM device.
2. Use the Solaris **pkgadd** program, to install the software by carrying out the following procedure:
   a. Type `pkgadd -d /cdrom/mq_solaris/`
   b. You are prompted for a list of components to install. Select those you require. If you want to install the entire MQSeries product, select **all**
   c. Press the Enter key

For further information on using **pkgadd** to install software packages, see the Solaris documentation.

Uninstalling the MQSeries for Sun Solaris, Intel Server

To uninstall MQSeries for Sun Solaris, Intel, follow the steps in "Migrating from an Earlier Version of MQSeries for Sun Solaris, Intel" on page 14.

Kernel Configuration

MQSeries makes use of semaphores, shared memory, and file descriptors, and it is probable that the **default** kernel configuration is not adequate.

In particular, the default number of semaphores is 60, which is *not* sufficient to support MQSeries.

If you attempt to use MQSeries without increasing `semmni`, the number of semaphores, the queue manager fails and produces a First Failure Support Technology™ (FFST) file. This file indicates that the system call `semop` received an argument that was not valid. An example of a possible set of actual kernel values on a Sun Solaris, Intel system is given in Figure 1. However, for Sun Solaris, Intel, the number of semaphores and semaphore sets has to be such that their control structures occupy less than 25% of the kernel storage.

After installation, you should review the machine's configuration. To do this type the following command:

```
sysdef -i
```

To change the values, add a `set parameter = value` line to the /etc/system file. For further information on setting up the system, see the Sun Solaris System Administration documentation.

```
set shmsys:shminfo_shmmax = 4294967295
set shmsys:shminfo_shmseg = 1024
set shmsys:shminfo_shmmin = 1
set shmsys:shminfo_shmmni = 1024
set shmsys:shminfo_shmem  = 1
set semsys:seminfo_sema   = 1
set semsys:seminfo_semaem = 16384
set semsys:seminfo_semvmx = 32767
set semsys:seminfo_semmni = 1024 (semmni < semmns)
set semsys:seminfo_semmap = 1026 (semmni +2)
set semsys:seminfo_semmns = 16384
set semsys:seminfo_semmsl = 100
set semsys:seminfo_semopm = 100
set semsys:seminfo_semmnu = 2048
set semsys:seminfo_semume = 256
set msgsys:msginfo_msgmni = 50
set msgsys:msginfo_msgmap = 1026
set msgsys:msginfo_msgmax = 4096
set msgsys:msginfo_msgmnb = 4096
set msgsys:msginfo_msgssz = 8
set msgsys:msginfo_msgtql = 40
set msgsys:msginfo_msgseg = 1024
set maxusers = 32
```

Figure 1. Kernel parameter values - example setting on a Sun Solaris, Intel system

Notes:

1. Shared memory usage does not vary with message rate or persistence.
2. Semaphore and swap usage does not vary with message size, message rate or persistence.
3. MQSeries queue managers are independent of each other. Therefore system kernel parameters, for example `shmmni`, `semmni`, `semmns`, and `semmnu` need to allow for the number of queue managers in the system.

For more details see the relevant SupportPac™, which is available on the MQSeries family home page. See "Available from the Internet" on page 53 for further details.

Sun Solaris, Intel has a low default system soft limit for the number of file descriptors. When running a multi-threaded process, you may reach the soft limit for file descriptors. This will give you the MQSeries reason code MQRC_UNEXPECTED_ERROR (2195), and an MQSeries FFST™ file.

To avoid this problem you can increase the system soft limit for the number of file descriptors. To do this:

- Edit the **/etc/system** file and change the value of the system soft limit to match the system hard limit (1024).

Additionally, if you are running MQSeries under the Lotus® Domino™ server, you can reduce the number of active server threads in the Domino HTTP server process. To do this:

- Open the server **Name and address** book, and reduce the **Number active threads** value on the server document to between 50 and 60.

Installing the Server and Client on the Same Machine

To install an MQSeries for Sun Solaris, Intel 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 MQSeries Clients CD-ROM.

You might install components from the MQSeries Clients CD-ROM onto 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.

Translated Messages

Messages in U.S. English are always available. If you require another of the languages that is supported by MQSeries for Sun Solaris, Intel, Version 5.1, 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
```

Translated Books

If you choose to install the Online Documentation component, you will get books in the language that was specified when your operating system was installed. However, some books may not be available in languages other than U.S. English and some hypertext links between books may not work. To overcome this you must choose to install a complete set of books in U.S. English as well as those in your national language. See "Viewing Online Books" on page 51 for more information about hypertext linking between translated books.

Verifying the Installation of MQSeries for Sun Solaris, Intel

This section describes how to verify that MQSeries for Sun Solaris, Intel has been correctly installed and configured. You do this by following the steps outlined in "Verification Procedure" on page 21.

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 the required communications protocols have been installed (and configured) on ***both*** machines.

The supported protocol is TCP/IP.

Note: The following examples assume that you will be using a TCP connection; for information about using other protocols, see the *MQSeries Intercommunication* book. However, you can also verify a *local* installation (which has no communications links with other MQSeries installations) without any communications protocols installed.

Verification Procedure

You can verify an MQSeries installation at three levels:

- A local (standalone) 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 23. For information on verifying a client/server installation, see the *MQSeries Clients* book.

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 Sun Solaris, Intel on the workstation (include the Base Server component as a minimum).
2. Create a default queue manager (in this example called `venus.queue.manager`):
   - At the command prompt in the window 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.

     Note: In prior releases of MQSeries it was necessary to run a script file called **amqscoma.tst** to define the MQSeries default objects. This step is not required in this release of the product.
3. Start the default queue manager:
   - Type `strmqm` and press Enter:

     A message tells you when the queue manager has started.
4. Enable MQSC commands by typing the following command and then 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 pressing Ctrl-D, or 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 following 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 then 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 the following directory, change to it now:
   `/opt/mqm/samp/bin`
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.

     Note: In prior releases of MQSeries it was necessary to run a script file called **amqscoma.tst** to define the MQSeries default objects. This step is not required in this release of the product.
2. Start the queue manager:
   - Type the following and then press Enter:
     `strmqm`

     A message tells you when the queue manager has started.
3. Enable MQSeries Commands (MQSC) by typing the following command and then 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 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 pressing Ctrl-D, or typing **end**, and pressing Enter.

   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 4 on page 25.

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. Find the process ID of the inetd with the command:
   ```
   ps -ef | grep inetd
   ```
4. Run the command:
   ```
   kill -1 inetd processid
   ```
5. 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.

     Note: In prior releases of MQSeries it was necessary to run a script file called **amqscoma.tst** to define the MQSeries default objects. This step is not required in this release of the product.
6. Start the queue manager:
   - Type the following and then press Enter:
     ```
     strmqm
     ```

     A message tells you when the queue manager has started.
7. 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.
8. Define a local queue (in this example, called `ORANGE.QUEUE`):
   - Type the following and press Enter:
     ```
     define qlocal (orange.queue)
     ```

     The message `MQSeries queue created` is displayed when the queue has been created.
9. 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`

10. 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.

User Exits

You ***must*** relink all your user exits with threaded libraries before you use them on this version of the product, to make them thread-safe.

See the *MQSeries Application Programming Guide* for further details on threaded libraries.

Setting the Queue Manager CCSID on MQSeries for Sun Solaris, Intel

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.

Examples of setting the CCSID:

```
export LANG=en_US.ISO8859-1
  uses the code set ISO8859-1
  and will set a CCSID of 819
export LANG=pl_PL.ISO8859-2
  uses the code set ISO8859-2
  and will set a CCSID of 912
```

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.

Note: The `ALTer QMGR CCSID` command is a new command supplied with MQSeries for Sun Solaris V5.1. See "Appendix F. Support for Different Code Sets on MQSeries for Sun Solaris, Intel" on page 79 for further information about supported code sets. See "Migration to euro Support" on page 82 for information on migrating to a CCSID that supports the euro character.

Queue Manager CCSID

Chapter 4. Planning to Install MQSeries Clients

This chapter is a summary of the hardware and software required to run the MQSeries for Sun Solaris, Intel client.

For information on how to install other MQSeries clients, see the *MQSeries Clients* book.

Applications on Version 5.1 clients

A Version 5 client can connect to all queue managers, non-Version 5 as well as Version 5. If you are connecting to a non-Version 5 queue manager you cannot use the new Version 5.1 features and structures in your MQSeries application on the client.

Sun Solaris, Intel Client: Hardware and Software Required

This section outlines the hardware and software requirements for an MQSeries client for Sun Solaris, Intel.

Hardware

An MQSeries client can run only on:

- Sun Intel desktop or server

with a minimum system disk space of 25 MB.

Note: Solaris systems from other manufacturers are not supported.

For connectivity, any communications hardware supporting TCP/IP.

Software

The following are prerequisites for MQSeries applications running on a Sun Solaris client.

IBM MQSeries for Sun Solaris, Intel, Version 5.1 clients run on:

- Sun Solaris 7 Intel Platform Edition Hardware 11/99, with patches 106981-10, 106542-10, 107637-03, and 107172-05

Connectivity

- TCP/IP as part of the base operating system

Hardware and Software for Sun Solaris, Intel

Compilers for MQSeries Applications on Sun Solaris, Intel Clients

The following compilers are supported:

- Sun WorkShop Compiler C, Version 5.0, with patches 107296-01, 107830-02, 107290-03, and 107361-04
- Sun WorkShop Compiler C++, Version 5.0, with patches 106328-08 and 107312-09

Client components

MQSeries Client
The MQSeries client code for your platform. It provides support to enable you to create and support your own applications and external applications.

Samples
Sample application programs.

MQSeries Client for Java
This allows Java applets running on your client machine to communicate with MQSeries. It includes security exits for encryption and authentication of messages sent across the Web by the MQSeries client for Java. These exits consist of some Java classes. To use the client for Java you need to have Java runtime code on your machine, at the Java Version 1.1.8-09a (Year 2000 compatible) level.

For information about Java runtime see the *MQSeries Using Java* book.

Note: If it is possible on your platform, at installation time the CLASSPATH environment variable will either get updated if already present or created if not.

MQSeries Internet Gateway documentation
MQSeries Internet Gateway documentation supplied in HTML format.

MQSeries Internet Gateway
Provides access to MQSeries applications via the Common Gateway Interface (CGI).

Chapter 5. Installing the MQSeries for Sun Solaris, Intel Client

Install the MQSeries for Sun Solaris, Intel client from the MQSeries product's server CD-ROM.

Before Installation

Before you can install an MQSeries client on your Sun Solaris, Intel system, you:

- Must create a group with the name `mqm`.
- Must create a user ID with the name `mqm`.
- Are recommended to create and mount a `/var/mqm` file system, or `/var/mqm`, `/var/mqm/log`, and `/var/mqm/errors` file systems.

  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.

After installation, this 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.

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.

Installation

Carry out the following procedure:

1. Check whether Volume Manager is running on your system by typing the following command:

   ```
   /usr/bin/ps -ef | /bin/grep vold
   ```

   If it is running, the CD is mounted on /cdrom/mqclient automatically. If it is not running, mount the CD by typing the following commands:

   ```
   mkdir -p /cdrom/mqclient
   mount -F hsfs -r /dev/dsk/cntndnsn /cdrom/mqclient
   ```

   substituting `cntndnsn` with the name of your CD-ROM device.

2. Use the Sun Solaris **pkgadd** program, to install the MQSeries client software by carrying out the following procedure:
   a. Type `pkgadd -d /cdrom/mqclient/solaris/mqs510.img`.
   b. You are prompted for a list of components to be installed. Select the ones you require - if you want to install all the components, select **all**.

      See "Client components" on page 30 for details.

      The component **MQSeries Client** for Java should be installed only if you have Java 1.1.8_09a (Year 2000 compatible) runtime code on your machine.
   c. Press the Enter key.

For further information on using **pkgadd** to install software packages, see the Sun Solaris documentation.

Uninstalling the MQSeries for Sun Solaris, Intel client

Use the Sun Solaris, Intel **pkgrm** command to uninstall the product. For further information on using pkgrm to uninstall software packages, see the Sun Solaris documentation.

Kernel Configuration

See the MQSeries family Web site for a SupportPac that gives additional performance information - see "Available from the Internet" on page 53.

Translated Messages

Messages in U.S. English are always available. If you require another of the languages that is supported by MQSeries for Sun Solaris, Intel, you *must* ensure that your NLSPATH environment variable includes the appropriate directory.

For example:

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

Removing an MQSeries Client from Sun Solaris, Intel

If you have previously installed MQSeries on your system, you need to remove the product using the **pkgrm** program.

If the product is present, but not installed correctly, you may need manually to delete the files and directories contained in /var/mqm and /opt/mqm.

Configuring and Verifying a Client

After you have installed the client, you need to configure your communications and verify the installation. For information on how to do this, see the *MQSeries Clients* book.

Chapter 6. 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 (see "Introducing 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, whilst 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

For 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.

MQSeries Command Sets

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, this command stops all queue manager code immediately:

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

Restarting a Queue Manager

To restart a queue manager called `saturn.queue.manager`, use the command:

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

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 38.

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" on page 43.
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 44.

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 42, 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 7. Using the MQSeries World Wide Web Interface

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

Overview of MQSeries Internet Gateway

MQSeries Internet Gateway provides a bridge between the synchronous World Wide Web and asynchronous MQSeries applications. With the 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 gateway is through HTML fill-out form POST requests; MQSeries applications respond by returning HTML pages to the gateway, via an MQSeries queue.

MQSeries Internet Gateway can be installed on AIX®, Compaq Tru64 UNIX, HP-UX, OS/2®, OS/390®, Sun Solaris (SPARC and Intel platforms), or Windows NT® systems.

The MQSeries Internet Gateway supports the CGI, ICAPI, ISAPI, and NSAPI Web server interfaces on all of these platforms, with the following exceptions:

- HP-UX does ***not*** support NSAPI.
- Compaq Tru64 UNIX and Sun Solaris (SPARC and Intel platforms) support CGI ***only***.

Obtaining More Information

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.

Obtaining 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 Web site.

The following Gateways are available:

- MQSeries Internet Gateway for AIX
- MQSeries Internet Gateway for HP-UX
- MQSeries Internet Gateway for OS/2
- MQSeries Internet Gateway for OS/390 OpenEdition®
- MQSeries Internet Gateway for Sun Solaris, Intel Platform Edition
- MQSeries Internet Gateway for Sun Solaris, SPARC Platform Edition
- MQSeries Internet Gateway for Windows NT

If you have downloaded a package from the web, go to the *Installing MQSeries Internet Gateway* documentation for a walkthrough of the installation procedure.

Chapter 8. Obtaining Additional Information

This chapter describes the sources of information that can be useful when you are using MQSeries for Sun Solaris, Intel or related products.

The MQSeries for Sun Solaris, Intel Library

The MQSeries for Sun Solaris, Intel Server CD-ROM contains the following documentation for MQSeries for Sun Solaris, Intel.

Order Number	Title	Description
	MQSeries Books	
GC34-5851	*MQSeries for Sun Solaris, Intel Platform Edition, Quick Beginnings*	Gives a brief overview of MQSeries for Sun Solaris, Intel and provides information on planning for, installing, and getting started with the product.
GC33-1349	*MQSeries Planning Guide*	Describes some key MQSeries concepts, identifies items that need to be considered before MQSeries is installed, including storage requirements, backup and recovery, security, and migration from earlier releases, and specifies hardware and software requirements for every MQSeries platform.
SC33-1872	*MQSeries Intercommunication*	Defines the concepts of distributed queuing, and explains how to set up a distributed queuing network in a variety of MQSeries environments. In particular, it demonstrates how to (1) configure communications to and from a representative sample of MQSeries products, (2) create required MQSeries objects, (3) create and configure MQSeries channels, and (4) establish MQSeries client/server connections. The use of channel exits is also described.
GC33-1632	*MQSeries Clients*	Describes how to install, configure, use, and manage MQSeries client systems.

Order Number	Title	Description
SC33-1873	*MQSeries System Administration*	Supports day-to-day management of local and remote MQSeries objects. It includes topics such as security, recovery and restart, transactional support, problem determination, and the dead-letter queue handler. It also includes the syntax of the MQSeries control commands.
SC33-1369	*MQSeries MQSC Command Reference*	Contains the syntax of the MQSC commands, which are used by MQSeries system operators and administrators to manage MQSeries objects.
SC33-1482	*MQSeries Programmable System Management*	Provides both reference and guidance information for users of MQSeries events, programmable command formats (PCFs), and installable services.
GC33-1876	*MQSeries Messages*	Describes "AMQ" messages issued by MQSeries. This book is available in softcopy only.
SC33-0807	*MQSeries Application Programming Guide*	Provides guidance information for users of the message queue interface (MQI). It describes how to design, write, and build an MQSeries application. It also includes full descriptions of the sample programs supplied with MQSeries.
SC33-1673	*MQSeries Application Programming Reference*	Provides comprehensive reference information for users of the MQI. It includes: data-type descriptions; MQI call syntax; attributes of MQSeries objects; return codes; constants; and code-page conversion tables.
SX33-6095	*MQSeries Application Programming Reference Summary*	Summarizes the information in the *MQSeries Application Programming Reference* manual.
SC33-1877	*MQSeries Using C++*	Provides both guidance and reference information for users of the MQSeries C++ programming-language binding to the MQI.
SC34-5390	*MQSeries Administration Interface Programming Guide and Reference*	Provides both guidance and reference information for users of the MQSeries Administration Interface.

Order Number	Title	Description
SC34-5349	*MQSeries Queue Manager Clusters*	Explains the concepts and terminology of MQSeries clustering, and shows how you can benefit by taking advantage of clusters. It summarizes the syntax of new and changed commands and shows a number of examples of tasks you can perform to set up and maintain clusters of queue managers.
SC34-5456	*MQSeries Using Java*	Provides both guidance and reference information for users of the MQSeries Bindings for Java and the MQSeries Client for Java.

Viewing Online Information

This section describes viewing:
- Online help
- Online books
- Books on the World Wide Web

Viewing Online Help

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

Viewing Online Books

Most of the MQSeries books are available in both hardcopy and softcopy formats.

The MQSeries online documentation is in Adobe Acrobat Portable Document Format (PDF) and HTML, on the server CD-ROM.

To view the MQSeries online documentation in HTML format directly from the CD-ROM, point your web browser to `/BOOKS/HTML/start.htm/` in the CD-ROM file system.

If you have installed the HTML-format documentation, point your web browser to:

```
/opt/mqm/books/html/start.htm
```

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. You can then follow links between the English books

only, but cannot link back into a translated book. In order to return to the translated book, you must use your browser's back button.

BookManager® Books

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®

HTML and PDF Books on the World Wide Web

The books listed in "The MQSeries for Sun Solaris, Intel Library" on page 49 are supplied as HTML and Portable Document Format on the MQSeries Server CD-ROM, and on the World Wide Web. For the latest information, see "Available from the Internet" on page 53.

Printing and Ordering Books

For information about ordering publications listed in this book, contact your IBM authorized dealer or marketing representative. 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).**

Printing Portable Document Format (PDF) Books

The MQSeries for Sun Solaris, Intel books are supplied in PDF format, which can be printed on a PostScript printer, on the MQSeries for Sun Solaris, Intel Server CD-ROM.

Table 2 on page 53 shows the file names used for the PDF files.

Note: The fifth character in the name represents the national language of the book. For example:

A U.S. English
F French
J Japanese
S Spanish

Table 2. MQSeries publications – file names

Book	File Name
MQSeries Planning Guide	CSQZAB03
MQSeries for Sun Solaris, Intel Platform Edition, Quick Beginnings	AMQ5AC00
MQSeries Intercommunication	CSQZAE03
MQSeries Clients	CSQZAF03
MQSeries System Administration	AMQZAG01
MQSeries Programmable System Management	CSQZAI03
MQSeries Command Reference	CSQZAJ03
MQSeries Application Programming Reference	CSQZAK03
MQSeries Application Programming Guide	CSQZAL03
MQSeries Application Programming Reference Summary	CSQZAM03
MQSeries Using C++	AMQZAN03
MQSeries Messages	AMQZAO01
MQSeries Administration Interface Programming Guide and Reference	CSQZAT01
MQSeries Queue Manager Clusters	CSQZAH01
MQSeries Using Java	CSQZAW04

Available from the Internet

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.

Appendix A. MQSeries for Sun Solaris, Intel, Version 5.1 at a glance

Program identification

Program name, IBM MQSeries for Sun Solaris, Intel, Version 5.1

Program number, 5639-L12

Hardware requirements

The MQSeries for Sun Solaris, Intel server code runs on any Intel based PC server or desktop system that is compatible with Sun Solaris 7 Intel platform edition. The minimum system disk space required is 25 MB. In addition, if you install the online books in HTML format you require 35 MB of storage in the /opt directory.

Software requirements

Sun Solaris Intel Platform Edition Hardware 11/99.

- MQSeries clients:

  Client code for Sun Solaris, Intel is distributed with the server code.

  Client software provides a remote interface to a LAN server. It may reside at the server or at a file server and be dynamically copied to the client for use, or it may reside on the client disk-space.

  Client support does not result in distributed coordination of units of work.

Connectivity

Any communications hardware supporting TCP/IP in the Sun Solaris, Intel environment may be used.

SNA is not supported on Sun Solaris, Intel Version 5.1. For the latest information about supported products, see the MQSeries family Web site:

`http://www.ibm.com/software/mqseries/platforms/supported.html`

Compilers supported

C++ programs can be compiled using the Sun WorkShop Compiler C++ Version 5.0.

C programs can be compiled using the Sun WorkShop Compiler C Version 5.0.

COBOL programs can be compiled using the Micro Focus COBOL for UNIX Version 4.1.20 compiler.

Java programs can be compiled using the Java Development Kit for Sun Solaris, Intel, Version 1.1.8_09a.

Delivery

MQSeries for Sun Solaris, Intel, Version 5.1 is delivered on CD-ROM.

Installation

MQSeries for Sun Solaris, Intel is installed using the Sun Solaris, Intel `pkgadd` command. Installation takes approximately 5 minutes.

Appendix B. Example installation process

Attention

Select options appropriate to your environment when the following prompt is displayed during the installation process:

```
Enter options to be installed [1-38,all,q?]
```

Figure 2 shows a typical MQSeries for Sun Solaris, Intel installation process.

```
The following packages are available:

1 mqm   MQSeries for Sun Solaris 2 (Intel) 5.1.0

Select package(s) you want to process (or 'all' to process
all packages). (default: all) [?,??,q]: all

MQSeries for Sun Solaris 2 (Intel) 5.1.0
Licensed Materials - Property of IBM

5639-L12
(C) Copyright International Business Machines Corp. 1994, 2000

All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.

Copyright (C) 1993-1995 Gradient Technologies Inc, Marlborough, Massachusetts

All rights reserved.

You do not appear to have a /var/mqm filesystem mounted.  It is
recommended that you create and mount separate /var/mqm and var/mqm/log
filesystems before installation.

Continue installation? [y,n,q]: y
```

Figure 2. An example MQSeries for Sun Solaris, Intel installation process (Part 1 of 3)

```
 1) MQSeries server
 2) Man pages
 3) Sample programs
 4) Sun Solaris 2 client libraries
 5) US English HTML publications
 6) French HTML publications
 7) German HTML publications
 8) Brazilian Portuguese HTML publications
 9) Japanese HTML publications
10) Korean HTML publications
11) Traditional Chinese HTML publications
12) Simplified Chinese HTML publications
13) Italian HTML publications
14) Spanish HTML publications
15) Java Bindings
16) Java Client
17) Java Base
18) US English Java documentation
19) Brazilian Portuguese Java documentation
20) Simplified Chinese Java documentation
21) German Java documentation
22) Spanish Java documentation
23) French Java documentation
24) Italian Java documentation
25) Japanese Java documentation
26) Korean Java documentation
27) Traditional Chinese Java documentation
28) Internet Gateway runtime
29) Internet Gateway samples
30) Spanish message catalog
31) French message catalog
32) German message catalog
33) Japanese message catalog
34) Italian message catalog
35) Brazilian Portuguese message catalog
36) Traditional Chinese message catalog
37) Simplified Chinese message catalog
38) Korean message catalog
```

Figure 2. An example MQSeries for Sun Solaris, Intel installation process (Part 2 of 3)

```
    Enter options to be installed [1-38,all,q?] all
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

The following files are being installed with setuid and/or setgid
permissions:
/opt/mqm/bin/amqcrsta_nd<setuid mqm setgid mqm>
/opt/mqm/bin/strmqcsv<setuid mqm setgid mqm>
   ......
   ......
Do you want to install these as setuid/setgid files [y,n,?,q] y

This package contains scripts that will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of <mqm> [y,n,?] y

Installing MQSeries for Sun Solaris 2 as <mqm>

##Installing part 1 of 1.
   ......
   ......
[verifying class <server>]
   ......
   ......
   ......
[verifying class <man>]
   ......
   ......
   ......
   ......
#Executing postinstall script.

Installation of <mqm> was successful.
```

Figure 2. An example MQSeries for Sun Solaris, Intel installation process (Part 3 of 3)

Appendix C. Sample MQI Programs and MQSC Files

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

Samples

Samples are provided as follows:

- "MQSC Command File Samples"
- "C and COBOL Program Samples"
- "Supporting Tuxedo for Transaction Processing" on page 63
- "Supporting Databases" on page 63
- "Miscellaneous Tools" on page 63

MQSC Command File Samples

Table 3 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 3. MQSC command files

File name	Purpose
amqscic0.tst	Defines objects for use in the sample CICS® transaction.
amqscos0.tst	Creates a set of MQI objects for use with the C and COBOL program samples.
amqmdefs.tst	Defines objects for the administration application sample.

C and COBOL Program Samples

Table 4 on page 62 lists the sample MQI source files. By default, the source files are **/opt/mqm**/samp and the compiled versions in directory **/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 4. Sample programs - source files

C	COBOL	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	amqmechx.cbl	Echoes a message from a message queue to the reply-to queue. Can be run as a triggered application program.
amqsgbr0.c	amq0gbr0.cbl	Writes messages from a queue to `stdout`, leaving the messages on the queue. Uses MQGET with the browse option.
amqsget0.c	amq0get0.cbl	Removes the messages from the named queue (using MQGET) and writes them to `stdout`.
amqsinqa.c	amqminqx.cbl	Reads the triggered queue; each request read as a queue name; responds with information about that queue.
amqsput0.c	amq0put0.cbl	Copies `stdin` to a message and then puts this message on a specified queue.
amqsreq0.c	amq0req0.cbl	Puts request messages on a specified queue and then displays the reply messages.
amqsseta.c	amqmsetx.cbl	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 Tuxedo for Transaction Processing

The samples include client transactions and some associated definitions and configuration files.

Table 5. Samples for transaction processing with Tuxedo

File name	Purpose
amqstxsx.c	Sample server
amqstxgx.c	Sample GET client application
amqstxpx.c	Sample PUT client application
amqstxvx.flds	Field definition
ubbstxcx.cfg	Configuration file

Supporting Databases

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

Table 6. Sample programs - databases

C	COBOL	Purpose
amqsxas0.c	amqsxas0.cbl	Updates a single database within an MQSeries unit of work.
amqsxag0.c	amqsxag0.cbl	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 formatter and code conversion.

Table 7. Miscellaneous files

File name	Location	Purpose
amqtrc.fmt	/opt/mqm/lib	Defines MQSeries trace formats.
ccsid.tbl	/var/mqm/conv/table	Edit this file to add any newly supported CCSID values to your MQSeries system.

Samples

Appendix D. Building applications on Sun Solaris, Intel

This appendix describes how to build application programs to run under Sun Solaris, Intel.

It contains the following sections:

Building applications in C on Sun Solaris, Intel

This section describes how to build application programs written in C to run under Sun Solaris, Intel.

C language include files

The MQSeries C include files are listed in Table 8. They are installed in the directory `/opt/mqm/inc/`. The include files are symbolically linked into `/usr/include`.

Table 8. C include files for MQSeries (Sun Solaris, Intel)

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

Preparing C programs

Work in your usual environment. Precompiled C programs are supplied in the `/opt/mqm/samp/bin` directory. MQSeries for Sun Solaris, Intel supports Workshop Compilers 5.0.

To compile, for example, the sample program amqsput0.c:

1. Export `LIB=/opt/mqm/lib;$LIB`
2. Ensure that the environment is set to use the correct versions of the compiler software and man pages:

```
export PATH=/opt/SUNWspro/bin:$PATH
export MANPATH=/opt/SUNWspro/man:/usr/man:$MANPATH
export LD_LIBRARY_PATH= \
/opt/SUNWspro/lib:$LD_LIBRARY_PATH
```

3. Compile the program (**the order in which you specify the libraries is important**):

```
$ cc -o<amqsput0><amqsput0>.c -mt -lmqm -lmqmcs  \
-lmqmzse -lsocket -lnsl -ldl
```

If you want to use the programs on a machine on which only the MQSeries client for Sun Solaris, Intel is installed, recompile the programs to link them with the client library. Following is an example instruction for building a client application. Again, the order of the parameters is important:

```
$ cc -o<amqsput0><amqsput0>.c -mt -lmqic -lmqmcs  \
-lmqmzse -lsocket -lnsl -ldl
```

Linking libraries

You need to link your programs with the MQSeries libraries that are appropriate for your application type.

Library file	Program or exit type
libmqm.so, libmqmcs.so, and libmqmzse.so	Server for C
libmqic.so, libmqmcs.so, and libmqmzse.so	Client for C

Notes:

1. If you are writing an installable service (as described in the *MQSeries Programmable System Management* book), you need to link to the libmqmzf.so library.
2. If you are producing an XA switch load file for external coordination by an XA-compliant transaction manager, for example, CICS, Transarc Encina, or Novell Tuxedo, link to the libmqmxa.a library.

Compiling data-conversion exits

On all platforms, the entry point to the module is MQStart.

This example shows how to compile a data-conversion exit program if your application is running in a nonthreaded environment, or is using POSIX V10 threading calls:

```
$ cc -c KPIC -I/opt/mqm/inc MYFORMAT.C

ld -G /opt/SUNWspro/SC5.0/lib/crtl.o \
/opt/SUNWspro/SC5.0/lib/crti.o  \
/opt/SUNWspro/SC5.0/lib/crtn.o  \
/opt/SUNWspro/SC5.0/lib/values-xt.o \
MYFORMAT.o -o MYFORMAT -lmqm -lthread -lsocket -lc -lnsl -ldl

$ cp MYFORMAT /var/mqm/exits
```

For more information about data-conversion exits, see the *MQSeries Application Programming Guide*.

Building applications in C++ on Sun Solaris, Intel

This section describes how to build application programs written in C++ to run under Sun Solaris, Intel.

For more information on using MQSeries with C++, see the *MQSeries Using C++* book.

C++ language include files

The MQSeries C++ include files are listed in Table 9. They are installed in the directory `/opt/mqm/inc/`. The include files are symbolically linked into `/usr/include`.

Table 9. C++ include files for MQSeries (Sun Solaris, Intel)

File name	Contents
<cmqc.h>	MQI data structures and manifest constants
<imqi.hpp>	C++ MQI classes (includes cmqc.h and imqtype.h)
<imqtype.h>	Defines the ImqBoolean data type
Note: The files are protected against multiple declaration, so you can include them many times.	

Preparing C++ programs

Work in your usual environment. MQSeries for Sun Solaris, Intel supports Sun Workshop Compiler C++ 5.0. Precompiled C++ programs are supplied in the `/opt/mqm/samp/bin/as` directory. Following is an example instruction for building the sample program imqsput0.cpp:

```
$ /opt/SUNWspro/bin/CC -D_SOLARIS -mt -library=iostream \
-o imqsput imqsput0.cpp -limqb23as -limqs23as -lmqm -lmqmcs -ldl \
-lnsl -lmqmzse -lsocket
```

If you want to use the programs on a machine on which only the MQSeries client for Sun Solaris, Intel is installed, recompile the programs to link them with the client library. Here is an example instruction for building the sample program imqsput0.ccp:

```
$ /opt/SUNWspro/bin/CC -D_SOLARIS -mt -library=iostream \
-o imqsput imqsput0.cpp -limqb23as -limqs23as -lmqic -lmqmcs -ldl \
-lnsl -lmqmzse -lsocket
```

Linking libraries

You must link with the MQSeries libraries that are appropriate for your application type:

Library files	Switches	Program or exit type
-limqb23as -limqs23as -lmqm -lmqmcs -lmqmzse -lsocket -lnsl -ldl	-mt -library=iostream	Server for C++
-limqb23as -limqs23as -lmqic -lmqmcs -lmqmzse -lsocket -lnsl -ldl	-mt -library=iostream	Client for C++

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

Building applications in COBOL

This section describes how to build application programs written in COBOL to run under Sun Solaris, Intel.

For more information on the MQSeries COBOL copy files, see the *MQSeries Application Programming Guide*.

Preparing COBOL programs

Work in your usual environment. MQSeries for Sun Solaris, Intel supports Micro Focus COBOL Developer Suite V4.1.20.

Precompiled COBOL programs are supplied in the `/opt/mqm/samp/bin` directory. Use the Micro Focus compiler from the directory `/opt/bin` to build a sample from source code.

To compile, for example, the sample program amq0put0:

1. Ensure that the environment is set. The following example assumes that COBOL is installed in the default directories:

   ```
   $ export COBDIR=/opt/lib/cobol
   $ export PATH=/usr/bin:$PATH
   $ export LD_LIBRARY_PATH=/$COBDIR/coblib:$LD_LIBRARY_PATH
   ```

2. Define the location of the copybooks that declare the MQI structures:

   ```
   $ export COBCPY="/opt/mqm/inc"
   ```

3. Compile the program:

   ```
   $ cob -vxP amq0put0.cbl -lmqmcb -lmqm -lmqmcs -lmqmzse
   ```

If you want to use the programs on a machine on which only the MQSeries client for Sun Solaris, Intel is installed, recompile the programs to link them with the client library libmqicb.so.

Linking libraries

You must link to one of the following libraries when building the application:

Library file	Program or exit type
libmqmcb.so	Server for COBOL
libmqicb.so	Client for COBOL

Installing MQSeries classes for Java

The MQSeries classes for Java can be installed from either the MQSeries Version 5.1 software Server CD, or Client CD.

If you choose the typical installation, the MQSeries classes for Java are included in the installation. If you choose to customize your installation, ensure that you check the MQSeries classes for Java option.

Notes:

1. If you want to use the native connection (bindings) mode, you **must** install from the server CD.
2. The MQSeries classes for Java files, documentation, and samples are installed in the directories shown in Table 10.

Table 10. MQSeries Classes for Java installation directories

Platform	File	Directory
Solaris	code documentation samples	opt/mqm/java/lib opt/mqm/html/mqjava opt/mqm/samp/javaclnt/langdir

MQSeries Java is contained in the following Java .jar files:

com.ibm.mq.jar
Code supporting all the connection options

com.ibm.mqbind.jar
Code supporting the bindings connection only

After installation, you will need to update your CLASSPATH environment variable to include the MQSeries Java code and samples directories.

A set of typical CLASSPATH settings for the Solaris platform is shown in the following text:

```
CLASSPATH=/opt/java1.1.1/lib/classes.zip: \
/opt/mqm/java/lib/com.ibm.mq.jar: \
/opt/mqm/java/lib/com.ibm.mqbind.jar: \
/opt/mqm/samp/javaclnt/en_us:
```

You also need to update the following environment variable:

```
LD_LIBRY_PATH=/opt/mqm/lib
```

Web server configuration

If you install MQSeries Java on a Web server, you can download and run MQSeries Java applications on machines that do not have MQSeries Java installed locally.

To make the MQSeries Java files accessible to your Web server, you must set up your Web server configuration to point to the directory where the client is installed. Consult your Web server documentation for details on how to configure the system.

Building applications in Java

This section describes how to build application programs written in Java to run under Sun Solaris, Intel.

Preparing Java programs

Make sure that your MQSeries Client for Java or MQSeries Bindings for Java installation directory is in your CLASSPATH environment variable. For example:

```
CLASSPATH=/opt/java1.1.1/lib/classes.zip:/opt/mqm/java/lib/com.ibm.mq.jar:
/opt/mqm/java/lib/com.ibm.mqbind.jar:/opt/mqm/samp/javaclnt/en_us:.
```

The following environment variable is required:

```
LD_LIBRARY_PATH=/opt/mqm/lib
```

To compile the class `MyClass.java`, for example, use the command:

```
$ javac MyClass.java
```

Note: If your MQSeries Java program handles large messages, you must increase the maximum Java heap size appropriately using the `-mx` option of the `java` command.

For more information on using Java with MQSeries, see the *MQSeries Using Java* book.

Building Tuxedo applications

Before you can run a TUXEDO application, you must build the server environment for MQSeries for Sun Solaris, Intel. It is assumed that you have a working TUXEDO environment.

1. Create a directory (for example <appdir>) in which the server environment is built and execute all commands in this directory.
2. Export the following environment variables, where TUXDIR is the root directory for TUXEDO:

```
$ export CFLAGS="-I/<APPDIR>"
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR/lib:/opt/mqm/lib:lib
$ export LD_LIBRARY_PATH=$(TUXDIR)/lib: \
/opt/mqm/lib:/lib$LD_LIBRARY_PATH
```

3. Add the following to the TUXEDO file TUXDIR/udataobj/RM

   Note: RM must include /opt/mqm/lib/libmqmcs and /opt/mqm/lib/libmqmzse.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
/opt/mqm/lib/libmqmxa.a /opt/mqm/lib/libmqm.so \
/opt/tuxedo/lib/libtux.so /opt/mqm/lib/libmqm.so \
/opt/mqm/lib/libmqmzse.so
```

4. Run the commands:

```
$ mkfldhdr amqstxvx.flds
$ viewc amqstxvx.v
$ buildtms -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f /opt/mqm/samp/amqstxsx.c  \
             -f /opt/mqm/lib/libmqm.so \
             -r MQSeries_XA_RMI -s MPUT1:MPUT \
             -s MGET1:MGET -v -bshm -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
             -f /opt/mqm/lib/libmqm.so \
             -r MQSeries_XA_RMI -s MPUT2:MPUT \
             -s MGET2:MGET -v -bshm -l -ldl
$ buildclient -o doputs -f amqstxpx.c \
             -f /opt/mqm/lib/libmqm.so \
             -f /opt/mqm/lib/libmqmzse.so \
             -f /opt/mqm/lib/libmqmcs.so
$ buildclient -o dogets -f amqstxgx.c \
             -f /opt/mqm/lib/libmqm.so \
             -f /opt/mqm/lib/libmqmzse.so \
             -f /opt/mqm/lib/libmqmcs.so
```

5. Edit the file ubbstxcx.cfg and add details of the machine name, working directories and queue manager as necessary. Execute the following command:

```
$ tmloadcf -y ubbstxcx.cfg
```

6. Create the TLOGDEVICE:

   ```
   $ tmadmin -c
   ```

   A prompt appears. At this point, enter:

   ```
   > crdl -z /<APPDIR>/TLOG1
   ```

7. Start the queue manager MYQUEUEMANAGER

   ```
   $ strmqm MYQUEUEMANAGER
   ```

8. Start the Tuxedo server:

   ```
   $ tmboot -y
   ```

You can now use the doputs and dogets programs to put messages to a queue and retrieve them from a queue.

For further information on the Tuxedo server environment, see the *MQSeries Application Programming Guide*.

Appendix E. Applying maintenance to MQSeries for Sun Solaris, Intel

This appendix tells you how to apply maintenance to MQSeries for Sun Solaris, Intel.

Maintenance updates in the form of a Program Temporary Fix (PTF) are supplied on CD-ROM. They can also be downloaded from:

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

Attention

Do not have any queue managers operating during installation of maintenance on MQSeries for Sun Solaris.

To end all running queue managers:

1. End the queue manager by issuing the command:

   ```
   endmqm -i QMgrName
   ```

2. Check that the queue manager has ended.

   Use the command:

   ```
   endmqm -w QMgrName
   ```

   The message returning should show that the queue manager is not available.

   Alternatively, use the command:

   ```
   ps -ef | grep mq
   ```

   where | is the pipe symbol. Check that there are no processes listed that are running command lines commencing `amq` or `runmq`. Ignore any that start with `amqi`.

3. Issue the **ipcs -a** command to identify any shared memory segments or semaphore sets that were created by MQSeries. Remove these using the **ipcrm** command.

Space Requirements

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

This allows a PTF to be removed, and the previous level to be automatically restored. If disk space is limited, the backup can be suppressed by creating an empty flag file called MQPTF_NOSAVE in the directory `/var/sadm/pkg`.

Note that if this option is used, the previous level will not be restored if a PTF is removed. The only way to restore a previous level in this instance is to reinstall the product and then to reapply a previous PTF image.

Applying the maintenance information

1. Mount the CD by typing the following commands:

   ```
   mkdir -p /cdrom/mq_solaris
   mount -F hsfs -r /dev/dsk/cntndnsn /cdrom/mq_solaris
   ```

   substituting `cntndnsn` with the name of your CD-ROM device.
2. Install the software by entering the following command:

   ```
   pkgadd -d /cdrom/mq_solaris/mqm/patchname
   ```

For further information on using **pkgadd** to install software packages, see the Solaris documentation.

Restoring the Previous Service Level

To restore the previous service level:

1. Log in as root, or use the command su.
2. Use the **pkgrm** command to remove the latest PTF from the system. For example, to remove PTF `U443859` issue the following command:

   ```
   pkgrm MQSERIES.U443859
   ```

   Error messages of the form `<shared pathname not removed>` can be ignored.

   Details of the **pkgrm** command can be found in the Solaris documentation, or by using the **man pkgrm** command.
3. If you have installed an MQI client, and the client was updated after installing the PTF that is being removed, then you ***must*** specifically update your MQI client installation again, after the PTF has been removed.

Appendix F. Support for Different Code Sets on MQSeries for Sun Solaris, Intel

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

If the locale is not set, the value of the LANG environment variable is used. If neither the locale nor LANG environment variable is set the CCSID used is 819 - the ISO 8859-1 code set.

Note: Not all the locales listed below are supported by all versions of Solaris.

See "Migration to euro Support" on page 82 for information on support for the euro character.

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 LC_CTYPE environment variable.

Table 11 shows the locales and the CCSIDs that are registered for the code set used by the locale.

Table 11. Locales and CCSIDs

Locale	Language	code set	CCSID
C	English	ISO 8859-1	819
ar ar_AA	Arabic Arabic	ISO 8859-6 ISO 8859-6	1089 1089
bu bu_BG	Bulgarian Bulgarian	ISO 8859-5 ISO 8859-5	915 915
cs cs_CZ	Czech Czech	ISO 8859-2 ISO 8859-2	912 912
da da_DK	Danish Danish	ISO 8859-1 ISO 8859-1	819 819
de de_DE	German German	ISO 8859-1 ISO 8859-1	819 819
de_AT	German - Austria	ISO 8859-1	819
de_CH	German - Switzerland	ISO 8859-1	819

Table 11. Locales and CCSIDs (continued)

Locale	Language	code set	CCSID
el el_GR	Greek Greek	ISO 8859-7 ISO 8859-7	813 813
en en_GB en_UK	English - United Kingdom English - United Kingdom English - United Kingdom	ISO 8859-1 ISO 8859-1 ISO 8859-1	819 819 819
en_AU	English - Australia	ISO 8859-1	819
en_CA	English - Canada	ISO 8859-1	819
en_US	English - USA	ISO 8859-1	819
es es_ES	Spanish Spanish	ISO 8859-1 ISO 8859-1	819 819
fi fi_FI	Finnish Finnish	ISO 8859-1 ISO 8859-1	819 819
fr fr_FR	French - France French - France	ISO 8859-1 ISO 8859-1	819 819
fr_BE	French - Belgium	ISO 8859-1	819
fr_CA	French - Canada	ISO 8859-1	819
fr_CH	French - Switzerland	ISO 8859-1	819
hr hr_HR	Croatian Croatian	ISO 8859-2 ISO 8859-2	912 912
hu hu_HR	Hungarian Hungarian	ISO 8859-2 ISO 8859-2	912 912
is is_IS	Icelandic Icelandic	ISO 8859-1 ISO 8859-1	819 819
it it_IT	Italian - Italy Italian - Italy	ISO 8859-1 ISO 8859-1	819 819
it_CH	Italian - Switzerland	ISO 8859-1	819
iw iw_IL	Hebrew Hebrew	ISO 8859-8 ISO 8859-8	916 916
ja ja_JP ja_JP.PCK	Japanese Japanese Japanese	eucJP eucJP PCK	5050 5050 943
ko ko_KR	Korean Korean	eucKR eucKR	970 970
mk mk_MK	Macedonian Macedonian	ISO 8859-5 ISO 8859-5	915 915
nl nl_NL	Dutch - Netherlands Dutch - Netherlands	ISO 8859-1 ISO 8859-1	819 819

Table 11. Locales and CCSIDs (continued)

Locale	Language	code set	CCSID
nl_BE	Dutch - Belgium	ISO 8859-1	819
no no_NO	Norwegian Norwegian	ISO 8859-1 ISO 8859-1	819 819
pl pl_PL	Polish Polish	ISO 8859-2 ISO 8859-2	912 912
POSIX	English	ISO 8859-1	819
pt pt_PT pt_BR	Portuguese Portuguese Portuguese-Brazil	ISO 8859-1 ISO 8859-1 ISO 8859-1	819 819 819
ro ro_RO	Romanian Romanian	ISO 8859-2 ISO 8859-2	912 912
ru ru_RU ru_SU	Russian Russian Russian	ISO 8859-5 ISO 8859-5 ISO 8859-5	915 915 915
sh sh_SP sh_YU	Serbocroatian Serbocroatian Serbocroatian	ISO 8859-2 ISO 8859-2 ISO 8859-2	912 912 912
sl sl_SL	Slovene Slovene	ISO 8859-2 ISO 8859-2	912 912
sk sk_SK	Slovak Slovak	ISO 8859-2 ISO 8859-2	912 912
sr sr_SP	Serbian Cyrillic Serbian Cyrillic	ISO 8859-5 ISO 8859-5	915 915
sv sv_SE	Swedish Swedish	ISO 8859-1 ISO 8859-1	819 819
tr tr_TR	Turkish Turkish	ISO 8859-9 ISO 8859-9	920 920
zh	Simplified Chinese	eucCN	1383
zh_TW	Traditional Chinese	eucTW	964
zh_TW.BIG5	Traditional Chinese	BIG5	950

For further information listing inter-platform support for these locales, see the *MQSeries Application Programming Reference* book.

Migration to euro Support

If you want to use the *euro* character with MQSeries, you should first install any operating system updates necessary to display the euro character.

Now modify your MQSeries system:

- Edit the existing CCSID.TBL file to enable the new euro version of the coded character set identifier (CCSID). To do this, remove the # symbol from the required line of the **CCSID Mapping** section of the CCSID.TBL file. When you have done this, all new queue managers you create will adopt the new euro CCSID.

  Note: If you want to create a new queue manager with a CCSID that supports the euro character, select a euro-supporting locale. For more information, refer to the MQSeries euro Web site at:

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

- To modify any existing queue managers that do not support the euro character, follow this procedure:

  1. Record the existing queue manager CCSID, with the MQSeries (MQSC) command:

     `DISplay QMGR CCSID`

  2. Change the CCSID to the euro support CCSID, with the MQSC command:

     `ALTer QMGR CCSID`

  3. Stop the queue manager.
  4. Restart the queue manager and any channels it uses.

  Note: The `ALTer QMGR CCSID` command is a new command supplied with MQSeries for Sun Solaris, Intel, V5.1.

  Now any new message issued using the queue manager CCSID uses the new euro CCSID. All messages now received using MQGET with conversion and requesting the queue manager CCSID to be used are converted into the euro CCSID. CCSIDs and object text (for example descriptions, definitions, and exit names), from existing messages are not changed.

Now modify your applications to support the euro character. If these use hard coded CCSIDs, ensure they now use the new euro CCSID.

Appendix G. 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

improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX	AS/400	BookManager
CICS	FFST	First Failure Support Technology
IBM	IBMLink	MQSeries
OpenEdition	OS/2	OS/390
SupportPac	TXSeries	VSE/ESA

Lotus and Domino are trademarks of Lotus Development Corporation in the United States, other countries, or both.

Intel is a registered trademark of Intel Corporation in the United States, other countries, or both.

Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java is a registered trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of X/Open Company Limited in the United States and other countries.

Other company, product, or service names may be the trademarks or service marks of others.

Index

R

S

T

U

V

W

Y

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.

GC34-5851-00