MQSeries® for Compaq Tru64 UNIX®

Quick Beginnings

Version 5.1

GC34-5684-00

MQSeries® for Compaq Tru64 UNIX®

Quick Beginnings

Version 5.1

GC34-5684-00

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix G. Notices" on page 97.

First edition (April 2000)

This edition applies to MQSeries for Compaq Tru64 UNIX Version 5 Release 1 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

Welcome to MQSeries for Compaq Tru64 UNIX

This book describes MQSeries for Compaq Tru64 UNIX and explains how to plan for, install, and use the product.

Note: This book describes IBM® MQSeries for Compaq Tru64 UNIX, Version 5.1. The previous version of this product is referred to as IBM MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX), Version 2 Release 2.1.

Road map

Table 1. Getting started road map

If you want to...	Refer to...
Read all about MQSeries for Compaq Tru64 UNIX on a single page	"Appendix A. MQSeries for Compaq Tru64 UNIX at a glance" on page 65
Learn about MQSeries	"Chapter 1. About MQSeries" on page 3
Learn about system requirements for MQSeries for Compaq Tru64 UNIX Server	"Chapter 2. Planning to install the MQSeries for Compaq Tru64 UNIX server" on page 13
Install MQSeries for Compaq Tru64 UNIX	"Chapter 3. Installing the MQSeries for Compaq Tru64 UNIX server" on page 19
Learn about system requirements for, and installing, the client	"Chapter 4. Installing the MQSeries client for Compaq Tru64 UNIX" on page 33
Start using command sets	"Chapter 5. Using the MQSeries command sets" on page 41
Learn about building applications	"Appendix D. Building applications on Compaq Tru64 UNIX" on page 75
Start using the Web Interface	"Chapter 6. Using the MQSeries World Wide Web interface" on page 53
View or print online documentation	"Chapter 7. Obtaining additional information" on page 55
Contact IBM	*Readers' Comment Form*

Conventions

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

- References to "Compaq Tru64 UNIX Version 4.0" include:
  DIGITAL UNIX Version 4.0D
  DIGITAL UNIX Version 4.0E
  Compaq Tru64 UNIX Version 4.0F
- **Boldface type** indicates the name of an item you need to select or the name of a command.
- *Italic 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.

Part 1. The IBM software servers

Chapter 1. About MQSeries

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 and 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.

Instrumentation events

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.

Part 2. Planning for and installing MQSeries for Compaq Tru64 UNIX

Chapter 2. Planning to install the MQSeries for Compaq Tru64 UNIX server

This chapter is a summary of the requirements for running MQSeries for Compaq Tru64 UNIX, the network protocols and the compilers supported, the delivery media, and the various components of the product.

Hardware requirements

- Any desktop or server system capable of running Compaq Tru64 UNIX.

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 them in the `/opt` directory.

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

Working data for MQSeries for Compaq Tru64 UNIX 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 Compaq Tru64 UNIX, Version 5.1 runs on the following versions:

- DIGITAL UNIX Version 4.0D or Version 4.0E
- Compaq Tru64 UNIX Version 4.0F or Version 5.0

Software requirements

For the latest information about supported products, see the MQSeries Family Web site:

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

Connectivity

The TCP/IP network protocol is supported.

- Any communications hardware supporting TCP/IP

For TCP/IP connectivity:

- TCP/IP is provided as part of the base operating system

For information on SNA support, see the README file supplied with MQSeries for Compaq Tru64 UNIX.

Compilers supported for Compaq Tru64 UNIX applications

The following compilers are supported:

- Compaq C for Tru64 UNIX (provided as part of the base operating system)
- Compaq C++ for Tru64 UNIX Version 6.2
- Micro Focus COBOL for UNIX Version 4.1B[1] or Version 4.1.00G[2]
- Java™ Development Kit for Compaq Tru64 UNIX, Version 1.1.8

Options

You may use the following options with MQSeries for Compaq Tru64 UNIX.

Transaction monitors

The following transaction processing monitor may be used:

- BEA Tuxedo Version 6.5

Coordination can be through the X/Open XA interface.

Databases

The following databases may be used:

- Oracle Release 7.3.4.0.0 or Oracle8i Release 8.1.5

Note: You must install a patch for the libaio_raw and libpthread libraries when using XA coordination with Oracle on DIGITAL UNIX Version 4.0D. The patch number is: OSF425-428.

1. Supplied by Compaq
2. Supplied by Micro Focus

Installation notes, and links to the patch itself, can be found on the Compaq Web page:

`http://ftp.support.compaq.com/public/unix/v4.0d/threadsafe_lib_v40d.html`

The patch is not required on DIGITAL UNIX Version 4.0E, Compaq Tru64 UNIX Version 4.0F, or Compaq Tru64 UNIX Version 5.0.

Delivery

MQSeries for Compaq Tru64 UNIX, V5.1 is supplied on CD-ROM.

Installation

MQSeries for Compaq Tru64 UNIX takes approximately 5 minutes to install, using the Compaq Tru64 UNIX installation program `setld` (with the `-l` option).

MQSeries for Compaq Tru64 UNIX components

When you install MQSeries for Compaq Tru64 UNIX you can choose which components to install. The components are as follows:

Runtime component
Support for external applications. This does **not** enable you to write your own applications.

Base Support to enable you to create and support your own applications. Requires the runtime component to be installed.

Server Support for client connections. Requires the runtime component to be installed.

MQSeries Client
MQSeries Client for Compaq Tru64 UNIX

- The MQSeries Client for Compaq Tru64 UNIX can be installed on the server machine, enabling you to have the MQSeries server and client on the same machine.

MQSeries for Java
MQSeries for Java has three components:

- Java base (required for client and bindings)
- Java client
- Java bindings

Notes:

1. 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.
2. Once 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.
3. 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 Compaq Tru64 UNIX in HTML format.

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

HTML and PDF versions of some of the MQSeries books are available in the following national languages:

- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Simplified Chinese
- Spanish
- Traditional Chinese
- U.S. English

MQSeries Internet Gateway
Provides access to MQSeries applications through HTML and CGI. The Internet Gateway has four components:
- Internet Gateway runtime
- Internet Gateway runtime data
- Internet Gateway samples
- Internet Gateway sample data

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

Samples
Sample application programs.

Note: The "base" and "runtime" components are mandatory and will be 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.

If you used amqscoma.tst to customize your settings for MQSeries Version 2, and you want to use the same settings with Version 5.1 of the product:
1. Save your copy of amqscoma.tst
2. Install MQSeries Version 5.1
3. Load your copy of amqscoma.tst and use the file to re-create your default objects

README file

Before starting to install MQSeries for Compaq Tru64 UNIX, review the README file, which you will find in the root directory of the appropriate server CD-ROM. There is one server CD-ROM for Compaq Tru64 UNIX Version 4.0 and one for Compaq Tru64 UNIX Version 5.0. The same README file is included on both.

Migrating from an earlier version of MQSeries for Compaq Tru64 UNIX

If you want to migrate from MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX), V2.2.1 to MQSeries for Compaq Tru64 UNIX, 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 45 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 re-create your MQSeries objects.
3. Uninstall the old MQSeries for Compaq Tru64 UNIX using the Compaq Tru64 UNIX program `setld` (with the `-d` option). 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 Compaq Tru64 UNIX, V5.1. See “Chapter 3. Installing the MQSeries for Compaq Tru64 UNIX server” on page 19 for information on installation.

Chapter 3. Installing the MQSeries for Compaq Tru64 UNIX server

This chapter tells you how to install MQSeries for Compaq Tru64 UNIX 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 20.

Note: There is one server CD-ROM for Compaq Tru64 UNIX Version 4.0 and one for Compaq Tru64 UNIX Version 5.0. An MQSeries server product is contained in the root directory of each.

See also "Installing the server and client on the same machine" on page 24.

Preparing for installation

This section guides you through some of the steps you must perform before you install MQSeries for Compaq Tru64 UNIX.

Before installation

Before you can install MQSeries for Compaq Tru64 UNIX you:

- Must install any required patches listed in the README
- 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.

Installation

This section describes the installation of the MQSeries for Compaq Tru64 UNIX server.

Notes:

1. If you have previously installed MQSeries on your system, you need to remove the product using the command: `setld -d`. See "Migrating from an earlier version of MQSeries for Compaq Tru64 UNIX" on page 17 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. Mount the appropriate CD-ROM by typing the following commands:
   a. For Compaq Tru64 UNIX Version 4.0:

      ```
      mount -r -t cdfs -o noversion,rrip /dev/rz18c /cdrom
      ```

      substituting the name of your CD-ROM device for *rz18c*.
   b. For Compaq Tru64 UNIX Version 5.0:

      ```
      mount -r -t cdfs -o noversion,rrip /dev/disk/cdrom0C /cdrom
      ```

      substituting the number of your CD-ROM device for *0C*.
2. Use the Compaq Tru64 UNIX program `setld` to install the software by carrying out the following procedure:
   a. Type `setld -l /cdrom`
   b. Press the Enter key.
   c. The installable subsets will be listed, and you enter the numbers of the MQSeries components you want to install. If you want to install the entire MQSeries product, select **all**.

      See "Appendix B. Example installation process" on page 67 for a screen dump of a typical install. See "MQSeries for Compaq Tru64 UNIX components" on page 15 for details.

For further information on using **setld** to install software packages, see the Compaq Tru64 UNIX documentation, or use the **man setld** command.

Uninstalling MQSeries server from Compaq Tru64 UNIX

If you have previously installed MQSeries on your system, you need to list the installed MQSeries components and then remove them.

List the MQSeries components using the Compaq Tru64 UNIX command:

```
setld -i | grep MQS
```

to list the *component name*s. You will see a list like this:

```
MQS_BASE         installed   IBM MQSeries Base subset
MQS_CLIENT       installed   IBM MQSeries Client subset
MQS_IGRUNTIME    installed   IBM MQSeries Internet Gateway runtime subset
MQS_IGRUNTIMED   installed   IBM MQSeries Internet Gateway runtime data
MQS_IGSAMPLE     installed   IBM MQSeries Internet Gateway samples subset
MQS_IGSAMPLED    installed   IBM MQSeries Internet Gateway sample data
MQS_JAVABASE     installed   IBM MQSeries Java Base subset
MQS_JAVABINDING  installed   IBM MQSeries Java Bindings subset
MQS_JAVACLIENT   installed   IBM MQSeries Java Client subset
MQS_LANG_DE_DE   installed   IBM MQSeries Language-German catalog
MQS_LANG_ES_ES   installed   IBM MQSeries Language-Spanish catalog
MQS_LANG_FR_FR   installed   IBM MQSeries Language-French catalog
MQS_LANG_IT_IT   installed   IBM MQSeries Language-Italian catalog
MQS_LANG_JA_JP   installed   IBM MQSeries Language-Japanese catalog
MQS_LANG_KO_KR   installed   IBM MQSeries Language-Korean catalog
MQS_LANG_PT_BR   installed   IBM MQSeries Language-Brazilian Portuguese catalog
MQS_LANG_ZH_CN   installed   IBM MQSeries Language-Simplified Chinese catalog
MQS_LANG_ZH_TW   installed   IBM MQSeries Language-Traditional Chinese catalog
MQS_MAN          installed   IBM MQSeries Manual pages
MQS_RUNTIME      installed   IBM MQSeries Runtime subset
MQS_SAMPLES      installed   IBM MQSeries Sample subset
MQS_SERVER       installed   IBM MQSeries Server subset
```

Remove the MQSeries components using the Compaq Tru64 UNIX command:

```
setld -d component name
```

Remove the MQS_BASE component *after* the other components have been uninstalled.

If the product installation had been manually altered, it may not completely uninstall, and you will need to manually delete the files and directories contained in `/opt/mqm`.

Kernel configuration

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

Semaphores

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

If you attempt to use MQSeries without increasing sem-mni, 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 Tru64 4.0 system is given in Figure 1.

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

```
dxkerneltuner
```

To change the ipc values, select the required fields and enter new values. For further information on setting up the system, see the Compaq Tru64 UNIX System Administration documentation.

V4.0

```
ipc:
  sem-mni=4096
  sem-msl=1000
  sem-opm=100
  sem-ume=1000
  shm-mni=4104
  shm-seg=1024
  num-of-sems=3000
  shm-max=2147483647
```

Figure 1. Kernel parameter values – minimum settings on a Tru64 4.0 system. These entries must be placed in the /etc/sysconfigtab file. You are recommended to perform this task using the dxkernel tuner.

V5.0

```
ipc:
  sem-mni=4096
  sem-msl=1000
  sem-opm=100
  sem-ume=1000
  shm-mni=4104
  shm-seg=1024
  shm-max=2147483647
```

Figure 2. Kernel parameter values – minimum settings on a Tru64 5.0 system. These entries must be placed in the /etc/sysconfigtab file. You are recommended to perform this task using the dxkernel tuner.

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 `shm-mni` and `sem-mni` need to allow for the total number of queue managers in the system.

Installing the server and client on the same machine

To install an MQSeries for Compaq Tru64 UNIX client on the server machine, use the appropriate MQSeries Server CD-ROM. Choose the client install option on the server CD-ROM to install the client code on the server machine.

You might install MQSeries client components on a machine and later want to install the MQSeries server component on the same machine. If so, you must remove the client components before you install the server, client, and any other components that you need.

You cannot install the server on a machine that already has client components installed on it.

Translated messages

Messages in U.S. English are always available. If you require another of the languages that is supported by MQSeries for Compaq Tru64 UNIX Version 5.1, or your message catalogues are in a nonstandard directory, you *must* ensure that your `NLSPATH` environment variable includes the appropriate directory.

Ensure that your `LANG` environment variable is set correctly. For example:

```
export LANG=fr_FR.ISO8859-1
```

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 “CD-ROM books directory” on page 58 for more information about hypertext linking between translated books.

Verifying the installation of MQSeries for Compaq Tru64 UNIX

This section describes how to verify that MQSeries for Compaq Tru64 UNIX has been correctly installed and configured. You do this by following the steps outlined in “Verification procedure” on page 25.

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.

For information on SNA support, see the README file supplied with MQSeries for Compaq Tru64 UNIX.

Note: The following examples assume that you will be using a TCP/IP 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 27. 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 Compaq Tru64 UNIX 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/IP address of the receiver workstation (note that this example is TCP/IP 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.

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
   ```

   This forces inetd to re-read the edited configuration file.

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 Compaq Tru64 UNIX

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 Compaq Tru64 UNIX V5.1. See “Appendix F. Support for different code sets on MQSeries for Compaq Tru64 UNIX” on page 93 for further information about supported code sets. See “Migration to euro support” on page 96 for information on migrating to a CCSID that supports the euro character.

Queue manager CCSID

Chapter 4. Installing the MQSeries client for Compaq Tru64 UNIX

This chapter summarizes the hardware and software required to run the MQSeries client for Compaq Tru64 UNIX, and tells you how to install the client.

Note: Both of the MQSeries server CD-ROMs (one for Compaq Tru64 UNIX Version 4.0 and one for Version 5.0) contain a copy of the MQSeries client. Use the appropriate server CD-ROM if you plan to install an MQSeries client and server on the same workstation, see "Installing the server and client on the same machine" on page 24.

The client is also available as a SupportPac™ to download from the Web. See the IBM MQSeries SupportPacs Web site for more information:

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

See the *MQSeries Clients* book for details of clients on other platforms.

Compaq Tru64 UNIX client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for Compaq Tru64 UNIX.

Hardware

An MQSeries client can run only on:

- Any desktop or server system capable of running Compaq Tru64 UNIX

with a minimum system disk space of 25 MB.

For connectivity, any communications hardware supporting TCP/IP.

For information on SNA support, see the README file supplied with MQSeries for Compaq Tru64 UNIX.

Software

The following are prerequisites for MQSeries applications running on a Compaq Tru64 UNIX client.

- Compaq Tru64 UNIX Version 4.0 or Version 5.0

Connectivity

- TCP/IP as part of the base operating system

For information on SNA support, see the README file supplied with MQSeries for Compaq Tru64 UNIX.

Compilers for MQSeries applications on Compaq Tru64 UNIX clients

The following compilers are supported:

- Compaq C for Tru64 UNIX (provided as part of the base operating system)
- Compaq C++ for Tru64 UNIX Version 6.2
- Micro Focus COBOL for UNIX Version 4.1B[3] or Version 4.1.00G[4]
- Java Development Kit for Compaq Tru64 UNIX, Version 1.1.8

Components for MQSeries for Compaq Tru64 UNIX client

MQSeries Client
The MQSeries client package for the Compaq Tru64 UNIX platform.

Samples
Sample application programs.

Runtime component
Support for external applications. This does **not** enable you to write your own applications.

Base Support to enable you to create and support your own applications. Requires the runtime component to be installed.

MQSeries Java Base
This is the base component for the MQSeries client for Java

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 following (or later compatible) levels:

Compaq Tru64 UNIX
Java Development Kit for Compaq Tru64 UNIX, Version 1.1.8

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.

3. Supplied by Compaq

4. Supplied by Micro Focus

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

MQSeries Internet Gateway
Provides access to MQSeries applications through HTML and CGI. The Internet Gateway has four components:
- Internet Gateway runtime subset
- Internet Gateway runtime data
- Internet Gateway samples subset
- Internet Gateway sample data

Installing on Compaq Tru64 UNIX

If you plan to install an MQSeries client and server on the same machine, you should use the appropriate MQSeries Server CD-ROM, see "Installing the server and client on the same machine" on page 24.

MQSeries clients for other platforms can be installed from the MQSeries Client CD-ROM. The Client CD-ROM also contains other MQSeries components that you might need to install on a workstation other than the server.

The MQSeries product is installed into the `/opt/mqm` directory. This ***cannot*** be changed.

Before installation

Before you can install an MQSeries client on your Compaq Tru64 UNIX 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` and `/var/mqm/errors` file systems.

  If you create separate partitions, the following directory ***must*** be on a local file system:
  – `/var/mqm`

  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. Mount the appropriate CD-ROM by typing the following commands:
   a. For Compaq Tru64 UNIX Version 4.0:

      ```
      mount -r -t cdfs -o noversion,rrip /dev/rz18c /cdrom
      ```

      substituting the name of your CD-ROM device for *rz18c*.
   b. For Compaq Tru64 UNIX Version 5.0:

      ```
      mount -r -t cdfs -o noversion,rrip /dev/disk/cdrom0C /cdrom
      ```

      substituting the number of your CD-ROM device for *0C*.
2. Use the Compaq Tru64 UNIX program `setld` to install the software by carrying out the following procedure:
   a. Type `setld -l /cdrom`
   b. Press the Enter key.
   c. You are prompted for a list of components to install. Select the ones you require, including MQSeries Client. If you want to install all the components, select **all**.

      The components you can select are:

      ```
      IBM MQSeries Base subset
      IBM MQSeries Client subset
      IBM MQSeries Internet Gateway runtime subset
      IBM MQSeries Internet Gateway runtime data
      IBM MQSeries Internet Gateway samples subset
      IBM MQSeries Internet Gateway sample data
      IBM MQSeries Java Base subset
      IBM MQSeries Java Client subset
      IBM MQSeries Manual pages
      IBM MQSeries Runtime subset
      IBM MQSeries Sample subset
      (The language catalog or catalogs of your choice.)
      ```

      See “Appendix B. Example installation process” on page 67 for a screen dump of a typical MQSeries Server install, which is similar.

      The component **MQSeries Client** for Java should be installed only if you have Java 1.1.8 (or later compatible) runtime code on your machine.

For further information on using **setld** to install software packages, see the Compaq Tru64 UNIX documentation, or use the **man setld** command.

Translated messages

Messages in U.S. English are always available. If you require another of the languages that is supported by MQSeries for Compaq Tru64 UNIX, or your message catalogues are in a nonstandard directory, you *must* ensure that your NLSPATH environment variable includes the appropriate directory.

Ensure that your LANG environment variable is set correctly. For example:

```
export LANG=fr_FR.ISO8859-1
```

Uninstalling an MQSeries client from Compaq Tru64 UNIX

If you have previously installed an MQSeries Client on your system, you can remove the product using the Compaq Tru64 UNIX commands:

```
setld -i
```

and

```
setld -d
```

List the MQSeries components using the Compaq Tru64 UNIX command:

```
setld -i | grep MQS
```

to list the *component name*s. You will see a list like this:

```
MQS_BASE         installed   IBM MQSeries Base subset
MQS_CLIENT       installed   IBM MQSeries Client subset
MQS_IGRUNTIME    installed   IBM MQSeries Internet Gateway runtime subset
MQS_IGRUNTIMED   installed   IBM MQSeries Internet Gateway runtime data
MQS_IGSAMPLE     installed   IBM MQSeries Internet Gateway samples subset
MQS_IGSAMPLED    installed   IBM MQSeries Internet Gateway sample data
MQS_JAVABASE     installed   IBM MQSeries Java Base subset
MQS_JAVABINDING  installed   IBM MQSeries Java Bindings subset
MQS_JAVACLIENT   installed   IBM MQSeries Java Client subset
MQS_LANG_DE_DE   installed   IBM MQSeries Language-German catalog
MQS_LANG_ES_ES   installed   IBM MQSeries Language-Spanish catalog
MQS_LANG_FR_FR   installed   IBM MQSeries Language-French catalog
MQS_LANG_IT_IT   installed   IBM MQSeries Language-Italian catalog
MQS_LANG_JA_JP   installed   IBM MQSeries Language-Japanese catalog
MQS_LANG_KO_KR   installed   IBM MQSeries Language-Korean catalog
MQS_LANG_PT_BR   installed   IBM MQSeries Language-Brazilian Portuguese catalog
MQS_LANG_ZH_CN   installed   IBM MQSeries Language-Simplified Chinese catalog
MQS_LANG_ZH_TW   installed   IBM MQSeries Language-Traditional Chinese catalog
MQS_MAN          installed   IBM MQSeries Manual pages
MQS_RUNTIME      installed   IBM MQSeries Runtime subset
MQS_SAMPLES      installed   IBM MQSeries Sample subset
MQS_SERVER       installed   IBM MQSeries Server subset
```

Issue the following command to remove a component:

```
setld -d component name
```

Remove the MQS_BASE component *after* the other components have been uninstalled.

If the product installation had been manually altered, it may not completely uninstall, and you will need to manually delete the files and directories contained in `/opt/mqm`.

Part 3. Using MQSeries for Compaq Tru64 UNIX

Chapter 5. 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

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

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

- The command name must be **crtmqm**, not **CRTMQM**.

- The flag must be -u, not -U.
- The dead-letter queue is SYSTEM.DEAD.LETTER.QUEUE.
- The argument is specified as jupiter.queue.manager, which is different from JUPITER.queue.manager.

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

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

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

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

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

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

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

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

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

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

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

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

endmqlsr
Ends a listener process.

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

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

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

rcrmqobj (re-create object)
Re-creates 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 commands (MQSC)

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

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

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

Running MQSC commands

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

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

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

PCF commands

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

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

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

Working with queue managers

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

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

Creating a default queue manager

The following command:

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

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

where:

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

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

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

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

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

Starting a queue manager

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

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

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

Stopping a queue manager

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

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

Quiesced shutdown

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

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

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

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

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

Immediate shutdown

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

For an immediate shutdown, the command is:

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

Preemptive shutdown

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

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

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

This stops all queue manager code immediately.

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 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 44.

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, see the *MQSeries Command Reference* book for the correct syntax.

Ending interactive input to MQSC

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

```
END
```

Alternatively, you can use the EOF character CTRL+D

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

Defining a local queue

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

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

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

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

The following MQSC command does this:

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

Notes:

1. Most of these attributes are the defaults as supplied with the product. However, they are shown here for purposes of illustration. You can omit them if you are sure that the defaults are what you want or have not been changed. See also "Displaying default object attributes".
2. USAGE (NORMAL) indicates that this queue is not an initiation queue or a transmission queue.
3. If you already have a local queue on the same queue manager with the name ORANGE.LOCAL.QUEUE, this command fails. Use the REPLACE attribute if you want to overwrite the existing definition of a queue, but see also "Changing local queue attributes" on page 50.

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 48, 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 6. Using the MQSeries World Wide Web interface

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

The MQSeries Internet Gateway is one of the installable components on the MQSeries server CD-ROMs.

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.

The MQSeries Internet Gateway supports the CGI Web server interface on the Compaq Tru64 UNIX platform.

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.

More information

Chapter 7. Obtaining additional information

This chapter describes the documentation for MQSeries for Compaq Tru64 UNIX, V5.1. It starts with a list of the publications, and then discusses:

MQSeries for Compaq Tru64 UNIX, V5.1 is described in the following books:

Table 2. MQSeries for Compaq Tru64 UNIX books

Order Number	Title	Description
	Compaq Tru64 UNIX Specific Book	
GC34-5684	*MQSeries for Compaq Tru64 UNIX, V5.1 Quick Beginnings*	Gives a brief overview of MQSeries for Compaq Tru64 UNIX, and provides information on planning for, installing, and getting started with the product.
	MQSeries Family Books	
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.

Table 2. MQSeries for Compaq Tru64 UNIX books (continued)

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 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.

Table 2. MQSeries for Compaq Tru64 UNIX books (continued)

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.

Hardcopy books

The book that you are reading now is *MQSeries for Compaq Tru64 UNIX Quick Beginnings*. This is the only book that is supplied in hardcopy with the product. However, all books listed in Table 2 on page 55 are available for you to order or print.

Note: MQSeries Messages is available in softcopy only.

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

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

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

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

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

For information about printing books, see “PDF” on page 58.

Online information

This section describes:

CD-ROM books directory

The books directory contains the MQSeries for Compaq Tru64 UNIX books in HTML and PDF formats. To access them point you Web browser to `books/start.htm`.

HTML

You can view the MQSeries online documentation in HTML format directly from the server CD-ROMs. All books except for the MQSeries Application Programming Reference Summary are available in U.S. English and also in some or all of the following national languages:

- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Spanish
- Simplified Chinese
- Traditional Chinese

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

PDF

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

The PDFs are available in U.S. English and also in some or all of the following national languages:

- Brazilian Portuguese
- French
- German

- Italian
- Japanese
- Korean
- Spanish
- Simplified Chinese
- Traditional Chinese

To find out which ones are available in your language, look for the appropriate directory on the CD-ROMs. The PDFs are in the directory books/pdf/ll_LL, where ll_LL is one of the following:

- pt_BR (Brazilian Portuguese)
- fr_FR (French)
- de_DE (German)
- it_IT (Italian)
- ja_JP (Japanese)
- ko_KR (Korean)
- es_ES (Spanish)
- zh_CN (Simplified Chinese)
- zh_TW (Traditional Chinese)
- en_US (U.S. English)

Within these directories, you can find the complete set of PDFs that are available. Table 3 shows the file names used for the PDF files.

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

B	Brazilian Portuguese	**F**	French
G	German	**I**	Italian
J	Japanese	**K**	Korean
S	Spanish	**Z**	Simplified Chinese
T	Traditional Chinese	**A**	U.S. English

Table 3. MQSeries publications – file names

Book	File Name
MQSeries Planning Guide	CSQZAB02
MQSeries for Compaq Tru64 UNIX, V5.1 Quick Beginnings	AMQ2AC00
MQSeries Intercommunication	CSQZAE02
MQSeries Clients	CSQZAF02
MQSeries System Administration	AMQZAG01
MQSeries Programmable System Management	CSQZAI02

Table 3. MQSeries publications – file names (continued)

Book	File Name
MQSeries Command Reference	CSQZAJ02
MQSeries Application Programming Reference	CSQZAK02
MQSeries Application Programming Guide	CSQZAL02
MQSeries Application Programming Reference Summary	CSQZAM02
MQSeries Using C++	AMQZAN02
MQSeries Messages	AMQZAO01
MQSeries Administration Interface Programming Guide and Reference	CSQZAT00
MQSeries Queue Manager Clusters	CSQZAH00
MQSeries Using Java	CSQZAW00

CD-ROM readme directory

The readme directory contains, in each national language, a file of up-to-date product information that you should read before installing or using this product.

There might also be a file of latest information (in U.S. English only) that became available after the readme files were translated.

HTML and PDF Books on the World Wide Web

The MQSeries for Compaq Tru64 UNIX books are available on the World Wide Web as well as on the server CD-ROMs. They are available in PDF and HTML format.

The MQSeries product family Web site is at:

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

By following links from this Web site you can:

- Obtain latest information about the MQSeries product family.
- Access the MQSeries books in HTML and PDF formats.
- Download MQSeries SupportPacs.

BookManager CD-ROMs

The MQSeries library is supplied in IBM BookManager® format on a variety of online library collection kits, including the *Transaction Processing and Data* collection kit, SK2T-0730.

You can view the softcopy books in IBM BookManager format using the following IBM licensed programs:

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

Online Help

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

Part 4. Appendixes

Appendix A. MQSeries for Compaq Tru64 UNIX at a glance

Program identification

Program name, IBM MQSeries for Compaq Tru64 UNIX, Version 5.1

Program number, 5765 E38

Hardware requirements

The MQSeries for Compaq Tru64 UNIX, V5.1 server code runs on any desktop or server system capable of running Compaq Tru64 UNIX. The minimum system disk space required is 25 MB.

Software requirements

MQSeries for Compaq Tru64 UNIX, V5.1 (server and client) runs under DIGITAL UNIX Version 4.0D, DIGITAL UNIX Version 4.0E, Compaq Tru64 UNIX Version 4.0F, or Compaq Tru64 UNIX Version 5.0

- MQSeries clients:

  Client code for Compaq Tru64 UNIX is distributed with the server code. It is also available as a SupportPac to download from the Web.

  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 Compaq Tru64 UNIX environment may be used.

For information on SNA support, see the README file supplied with MQSeries for Compaq Tru64 UNIX.

Compilers supported

Application programs can be written in C++, C, COBOL or Java. C++ programs can be compiled using the Compaq C++ for Tru64 UNIX Version 6.2 compiler. C programs can be compiled using the Compaq C for Tru64 UNIX compiler provided as part of the base operating system. COBOL programs can be compiled using the Micro Focus COBOL for UNIX Version 4.1B[5] or Micro Focus COBOL for UNIX Version 4.1.00G[6] compilers. Java programs can be compiled using the Java Development Kit for Compaq Tru64 UNIX, Version 1.1.8.

5. Supplied by Compaq

6. Supplied by Micro Focus

Delivery

MQSeries for Compaq Tru64 UNIX, V5.1 is delivered on two CD-ROMs: one for Compaq Tru64 UNIX Version 4.0 and one for Compaq Tru64 UNIX Version 5.0.

Installation

MQSeries for Compaq Tru64 UNIX is installed using the Compaq Tru64 UNIX `setld` command. Installation takes approximately 5 minutes.

Appendix B. Example installation process

Figure 3 shows a typical MQSeries for Compaq Tru64 UNIX installation process.

```
Licensed Materials - Property of IBM

5765-E38
(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.

Working....Mon Feb 14 18:28:19 GMT+0500 2000

*** Enter subset selections ***

The following subsets are mandatory and will be installed automatically
unless you choose to exit without installing any subsets:

      * IBM MQSeries Base subset
      * IBM MQSeries Runtime subset

The subsets listed below are optional:

     There may be more optional subsets than can be presented on a single
     screen. If this is the case, you can choose subsets screen by screen
     or all at once on the last screen. All of the choices you make will
     be collected for your confirmation before any subsets are installed.

     1) IBM MQSeries Client subset

--- MORE TO FOLLOW ---
Enter your choices or press RETURN to display the next screen.

Choices (for example, 1 2 4-6):
     2) IBM MQSeries Internet Gateway runtime data
     3) IBM MQSeries Internet Gateway runtime subset
     4) IBM MQSeries Internet Gateway sample data
     5) IBM MQSeries Internet Gateway samples subset
     6) IBM MQSeries Java Base subset
     7) IBM MQSeries Java Bindings subset
     8) IBM MQSeries Java Client subset
```

Figure 3. An example MQSeries for Compaq Tru64 UNIX installation process (Part 1 of 3)

```
     9) IBM MQSeries Language-Brazilian Portuguese catalog
    10) IBM MQSeries Language-French catalog
    11) IBM MQSeries Language-German catalog
    12) IBM MQSeries Language-Italian catalog
    13) IBM MQSeries Language-Japanese catalog
    14) IBM MQSeries Language-Korean catalog
    15) IBM MQSeries Language-Simplified Chinese catalog
    16) IBM MQSeries Language-Spanish catalog
    17) IBM MQSeries Language-Traditional Chinese catalog

--- MORE TO FOLLOW ---
Enter your choices or press RETURN to display the next screen.

Choices (for example, 1 2 4-6):
    18) IBM MQSeries Manual pages
    19) IBM MQSeries Sample subset
    20) IBM MQSeries Server subset

The following choices override your previous selections:

    21) ALL mandatory and all optional subsets
    22) MANDATORY subsets only
    23) CANCEL selections and redisplay menus
    24) EXIT without installing any subsets

Add to your choices, choose an overriding action or
press RETURN to confirm previous selections.

Choices (for example, 1 2 4-6):  21
You are installing the following mandatory subsets:

        IBM MQSeries Base subset
        IBM MQSeries Runtime subset

You are installing the following optional subsets:

        IBM MQSeries Client subset
        IBM MQSeries Internet Gateway runtime data
        IBM MQSeries Internet Gateway runtime subset
        IBM MQSeries Internet Gateway sample data
        IBM MQSeries Internet Gateway samples subset
        IBM MQSeries Java Base subset
        IBM MQSeries Java Bindings subset
        IBM MQSeries Java Client subset
        IBM MQSeries Language-Brazilian Portuguese catalog
        IBM MQSeries Language-French catalog
        IBM MQSeries Language-German catalog
```

Figure 3. An example MQSeries for Compaq Tru64 UNIX installation process (Part 2 of 3)

```
Press RETURN to display the next screen:
        IBM MQSeries Language-Italian catalog
        IBM MQSeries Language-Japanese catalog
        IBM MQSeries Language-Korean catalog
        IBM MQSeries Language-Simplified Chinese catalog
        IBM MQSeries Language-Spanish catalog
        IBM MQSeries Language-Traditional Chinese catalog
        IBM MQSeries Manual pages
        IBM MQSeries Sample subset
        IBM MQSeries Server subset

Is this correct? (y/n):
Checking file system space required to install selected subsets:

File system space checked OK.

22 subset(s) will be installed.

Loading 1 of 22 subset(s)....

IBM MQSeries Base subset
   Copying from . (disk)
   Verifying

Loading 2 of 22 subset(s)....

IBM MQSeries Java Base subset
   Copying from . (disk)
   Verifying

Loading 3 of 22 subset(s)....

IBM MQSeries Server subset
   Copying from . (disk)
   Verifying

.
.
.
```

Figure 3. An example MQSeries for Compaq Tru64 UNIX installation process (Part 3 of 3)

Appendix C. Sample MQI programs and MQSC files

MQSeries for Compaq Tru64 UNIX provides a set of short sample MQI programs and MQSC command files. You can use these directly or modify them for experimental purposes.

MQSC command file samples

Table 4 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 4. MQSC command files

File name	Purpose
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 5 lists the sample MQI source files. By default, the source files are in directory `/opt/mqm/samp` and the compiled versions are 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 5. 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`.

Table 5. Sample programs - source files (continued)

C	COBOL	Purpose
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 6. 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

By default, the database samples are located in directory `/opt/mqm/samp/xatm`.

Table 7. Sample programs - databases

C	COBOL	Purpose
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 8. 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.

MQSC samples

Appendix D. Building applications on Compaq Tru64 UNIX

This appendix describes how to build application programs to run under Compaq Tru64 UNIX.

It contains the following sections:

Throughout this chapter the \ character is used to split long commands over more than one line. Do not enter this character, enter each command as a single line.

Building applications in C on Compaq Tru64 UNIX Version 4.0

This section describes how to build application programs written in C to run under Compaq Tru64 UNIX Version 4.0.

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 (Compaq Tru64 UNIX Version 4.0)

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. Here is an example instruction for building the sample program amqsput0.c in a nonthreaded environment:

```
$ cc -std1 -o amqsput amqsput0.c -lmqm
```

Here is an example instruction for building the sample program amqsput0.c in a threaded environment:

```
$ cc -std1 -pthread -o amqsput amqsput0.c -lmqm_r
```

If you want to use the programs on a machine on which only the MQSeries client for Compaq Tru64 UNIX is installed, recompile the programs to link them with the client library. Here is an example instruction for building a nonthreaded client:

```
$ cc -std1 -o amqsput amqsput0.c -lmqmic
```

Here is an example instruction for building a threaded client:

```
$ cc -std1 -pthread -o amqsput amqsput0.c -lmqmic_r
```

Linking libraries

In a nonthreaded environment, you must link to one of the following libraries:

Library file	Program or exit type
libmqm.so	Server for C
libmqic.so	Client for C

In a threaded environment, you must link to one of the following libraries:

Library file	Program or exit type
libmqm_r.so	Server for C
libmqic_r.so	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.

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 in a nonthreaded environment:

```
$ cc -std1 -c -I /opt/mqm/inc myformat.c
$ cc -std1 -shared -o myformat myformat.o -L /opt/mqm/lib -lmqm -e MQStart -lc
$ cp myformat /var/mqm/exits
```

This example shows how to compile a data-conversion exit program in a threaded environment:

```
$ cc -std1 -c -I /opt/mqm/inc myformat.c
$ cc -std1 -shared -pthread -o myformat_r myformat.o -L /opt/mqm/lib \
  -lmqm_r -e MQStart -lc
$ cp myformat /var/mqm/exits
```

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

Building applications in C on Compaq Tru64 UNIX Version 5.0

This section describes how to build application programs written in C to run under Compaq Tru64 UNIX Version 5.0.

> **Attention**
> Only one set of libraries is supplied with MQSeries for Compaq Tru64 UNIX, V5.1, on Compaq Tru64 UNIX Version 5.0. You can build threaded or nonthreaded applications by linking with these libraries.

C language include files

The MQSeries C include files are listed in Table 10. They are installed in the directory `/opt/mqm/inc/`. The include files are symbolically linked into `/usr/include`.

Table 10. C include files for MQSeries (Compaq Tru64 UNIX Version 5.0)

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. Here is an example instruction for building the sample program amqsput0.c:

```
$ cc -std1 -o -pthread amqsput amqsput0.c -lmqm
```

If you want to use the programs on a machine on which only the MQSeries client for Compaq Tru64 UNIX is installed, recompile the programs to link them with the client library. Here is an example instruction for building a client:

```
$ cc -std1 -o -pthread amqsput amqsput0.c -lmqmic
```

Linking libraries

You must link to one of the following libraries:

Library file	Program or exit type
libmqm.so	Server for C
libmqic.so	Client for C

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:

```
$ cc -std1 -c -I /opt/mqm/inc myformat.c
$ cc -std1 -shared -pthread -o myformat myformat.o -L /opt/mqm/lib \
  -lmqm -e MQStart -lc
$ cp myformat /var/mqm/exits
```

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

Building applications in C++ on Compaq Tru64 UNIX Version 4.0

This section describes how to build application programs written in C++ to run under Compaq Tru64 UNIX Version 4.0.

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 11. They are installed in the directory `/opt/mqm/inc/`. The include files are symbolically linked into `/usr/include`.

Table 11. C++ include files for MQSeries (Compaq Tru64 UNIX Version 4.0)

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. Precompiled C++ programs are supplied in the `/opt/mqm/samp/bin/ff` directory. Here is an example instruction for building the sample program imqsput0.cpp in a nonthreaded environment:

```
$ cxx -std1 -o imqsput imqsput0.cpp -limqs23ff -limqb23ff
```

Here is an example instruction for building the sample program imqsput0.cpp in a threaded environment:

```
$ cxx -std1 -pthread -o imqsput imqsput0.cpp -limqs23ff_r -limqb23ff_r
```

If you want to use the programs on a machine on which only the MQSeries client for Compaq Tru64 UNIX is installed, recompile the programs to link them with the client library. Here is an example instruction for building the sample program imqsput0.ccp as a nonthreaded client:

```
$ cxx -std1 -o imqsput imqsput0.cpp -limqc23ff -limqb23ff
```

Here is an example instruction for building the sample program imqsput0.cpp as a threaded client:

```
$ cxx -std1 -pthread -o imqsput imqsput0.cpp -limqc23ff_r -limqb23ff_r
```

Linking libraries

In a nonthreaded environment, you must link to one of the following libraries:

Library file	Program or exit type
libimqs23ff.so	Server for C++
libimqc23ff.so	Client for C++

In a threaded environment, you must link to one of the following libraries:

Library file	Program or exit type
libimqs23ff_r.so	Server for C++
libimqc23ff_r.so	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.

Compiling data-conversion exits on Compaq Tru64 UNIX

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

This example shows how to compile a data-conversion exit program on Compaq Tru64 UNIX:

```
$ cxx -c -I /opt/mqm/inc MYFORMAT.CPP
$ cxx -shared MYFORMAT.o -shared -o MYFORMAT -L /opt/mqm/lib \
  -lc -limqb23ff -limqs23ff -lmqm -e MQStart
$ cp MYFORMAT /opt/mqm/lib
```

Building applications in C++ on Compaq Tru64 UNIX Version 5.0

This section describes how to build application programs written in C++ to run under Compaq Tru64 UNIX Version 5.0.

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 12. They are installed in the directory `/opt/mqm/inc/`. The include files are symbolically linked into `/usr/include`.

Table 12. C++ include files for MQSeries (Compaq Tru64 UNIX Version 5.0)

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. Precompiled C++ programs are supplied in the `/opt/mqm/samp/bin/ff` directory. Here is an example instruction for building the sample program imqsput0.cpp:

```
$ cxx -pthread -o imqsput imqsput0.cpp -limqs23ff -limqb23ff
```

If you want to use the programs on a machine on which only the MQSeries client for Compaq Tru64 UNIX is installed, recompile the programs to link them with the client library. Here is an example instruction for building the sample program imqsput0.cpp as a client:

```
$ cxx -pthread -o imqsput imqsput0.cpp -limqc23ff -limqb23ff
```

Linking libraries

You must link to one of the following libraries:

Library file	Program or exit type
libimqs23ff.so	Server for C++
libimqc23ff.so	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.

Compiling data-conversion exits on Compaq Tru64 UNIX

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

This example shows how to compile a data-conversion exit program on Compaq Tru64 UNIX:

```
$ cxx -c -I /opt/mqm/inc MYFORMAT.CPP
$ cxx -shared MYFORMAT.o -shared -o MYFORMAT -L /opt/mqm/lib \
  -lc -limqb23ff -limqs23ff -lmqm -e MQStart
$ cp MYFORMAT /opt/mqm/lib
```

Building applications in COBOL

This section describes how to build application programs written in COBOL to run under Compaq Tru64 UNIX.

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

Preparing COBOL programs

Use the Micro Focus COBOL compiler to compile your programs. The copy files which declare the structures are in the `/opt/mqm/inc` directory.

See the Micro Focus COBOL compiler documentation for a description of the environment variables that you must set up.

```
$ export COBDIR=/opt/cobol
$ export COBCPY=/opt/mqm/inc
$ export LD_LIBRARY_PATH=/opt/mqm/lib:$LD_LIBRARY_PATH
```

Here is an example instruction for building the sample program amq0put0.cbl in a nonthreaded environment:

```
$ cob -vxP amq0put0.cbl -lmqmcb
```

Here is an example instruction for building the sample program amq0put0.cbl in a threaded environment:

```
$ cob -vxP amq0put0.cbl -lmqmcb_r
```

If you want to use the programs on a machine on which only the MQSeries client for Compaq Tru64 UNIX is installed, recompile the programs to link them with the client library. Here is an example instruction for building a nonthreaded client:

```
$ cob -vxP amq0put0.cbl -lmqmicb
```

Here is an example instruction for building a threaded client:

```
$ cob -vxP amq0put0.cbl -lmqmicb_r
```

Linking libraries

You must link to one of the following libraries:

Library file	Program or exit type
libmqmcb.so	Server for COBOL
libmqmicb.so	Client for COBOL
libmqmcb_r.so	Threaded applications

Building applications in Java

This section describes how to build application programs written in Java to run under Compaq Tru64 UNIX.

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=/usr/opt/jdk118/lib/classes.zip:/opt/mqm/java/lib: \
          /opt/mqm/samp/mqbind/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 Compaq Tru64 UNIX. The procedure is the same on Compaq Tru64 UNIX Version 4.0 and Version 5.0. 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:

```
$ TUXDIR=/TUXDIR; export TUXDIR
$ PATH=$TUXDIR/bin:$PATH; export PATH
$ COBCPY=:$TUXDIR/cobinclude; export COBCPY
$ COBOPT="-C ANS85 -C ALIGN=8 -C NOIBMCOMP -C TRUNC=ANSI \
         -C OSEXT=cbl"; export COBOPT
$ LD_LIBRARY_PATH=$TUXDIR/lib:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH
```

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

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic:-lmqm -lmqmzse \
-lmqmxa -lmqmcs -L/$TUXDIR/lib -ltux
```

4. Run the commands:

```
$ mkfldhdr /opt/mqm/samp/amqstxvx.flds
$ viewc /opt/mqm/samp/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
$ buildserver -o MQSERV2 -f /opt/mqm/samp/amqstxsx.c \
              -f /opt/mqm/lib/libmqm.so -r MQSeries_XA_RMI \
              -s MPUT2:MPUT -s MGET2:MGET -v -bshm
$ buildclient -o doputs -f /opt/mqm/samp/amqstxpx.c \
              -f /opt/mqm/lib/libmqm.so -f /opt/mqm/lib/libmqmzse.so \
              -f /opt/mqm/lib/libmqmcs.so
$ buildclient -o dogets -f /opt/mqm/samp/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 then 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 Compaq Tru64 UNIX

This appendix tells you how to apply maintenance to MQSeries for Compaq Tru64 UNIX.

Maintenance updates in the form of a Program Temporary Fix (PTF) can be downloaded from:

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

They are also available on CD-ROM.

Before starting to apply maintenance to MQSeries for Compaq Tru64 UNIX, review the README file supplied with the product.

Attention

Do not have any queue managers operating during installation of maintenance on MQSeries for Compaq Tru64 UNIX.

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.

Applying maintenance

For a PTF from the Web site:

1. Files typically have names like *u123456*. Download the file into a temporary directory, and make that your working directory. In the following examples, substitute the name of the file you download for *u123456*.
2. Uncompress the file with the command:

   ```
   uncompress u123456
   ```

3. Untar the file if necessary with the command:

   ```
   tar -xvf u123456
   ```

4. Install the software with the command:

   ```
   setld -l .
   ```

For a PTF on CD-ROM:

1. Mount the CD-ROM by typing the following commands:
   a. For Compaq Tru64 UNIX Version 4.0:

      ```
      mount -r -t cdfs -o noversion,rrip /dev/rz18c /cdrom
      ```

      substituting the name of your CD-ROM device for *rz18c*.
   b. For Compaq Tru64 UNIX Version 5.0:

      ```
      mount -r -t cdfs -o noversion,rrip /dev/disk/cdrom0C /cdrom
      ```

      substituting the number of your CD-ROM device for *0C*.
2. Install the software by entering the following command:

   ```
   setld -l /cdrom/PTF-dir
   ```

For further information on using **setld** to install software packages, see the Compaq Tru64 UNIX documentation, or use the **man setld** command.

Restoring the previous service level

To restore the previous service level:

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

   ```
   setld -d MQS_U443859
   ```

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

   Details of the **setld** command can be found in the Compaq Tru64 UNIX documentation, or by using the **man setld** 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.

Space requirements

Appendix F. Support for different code sets on MQSeries for Compaq Tru64 UNIX

MQSeries for Compaq Tru64 UNIX 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 MQSeries for Compaq Tru64 UNIX.

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 ISO8859-1 code set.

Note: Not all the locales listed below are supported by all versions of Compaq Tru64 UNIX.

See "Migration to euro support" on page 96 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 13 shows the locales and the CCSIDs that are registered for the code set used by the locale.

Table 13. Locales and CCSIDs

Locale	Language	code set	CCSID
C	English	ISO8859-1	819
ar ar_AA	Arabic Arabic	ISO8859-6 ISO8859-6	1089 1089
bu bu_BG	Bulgarian Bulgarian	ISO8859-5 ISO8859-5	915 915
cs cs_CZ	Czech Czech	ISO8859-2 ISO8859-2	912 912
da da_DK	Danish Danish	ISO8859-1 ISO8859-1	819 819
de de_DE	German German	ISO8859-1 ISO8859-1	819 819
de_AT	German - Austria	ISO8859-1	819
de_CH	German - Switzerland	ISO8859-1	819

Table 13. Locales and CCSIDs (continued)

Locale	Language	code set	CCSID
el el_GR	Greek Greek	ISO8859-7 ISO8859-7	813 813
en en_GB en_UK	English - United Kingdom English - United Kingdom English - United Kingdom	ISO8859-1 ISO8859-1 ISO8859-1	819 819 819
en_AU	English - Australia	ISO8859-1	819
en_CA	English - Canada	ISO8859-1	819
en_US	English - USA	ISO8859-1	819
es es_ES	Spanish Spanish	ISO8859-1 ISO8859-1	819 819
fi fi_FI	Finnish Finnish	ISO8859-1 ISO8859-1	819 819
fr fr_FR	French - France French - France	ISO8859-1 ISO8859-1	819 819
fr_BE	French - Belgium	ISO8859-1	819
fr_CA	French - Canada	ISO8859-1	819
fr_CH	French - Switzerland	ISO8859-1	819
hr hr_HR	Croatian Croatian	ISO8859-2 ISO8859-2	912 912
hu hu_HR	Hungarian Hungarian	ISO8859-2 ISO8859-2	912 912
is is_IS	Icelandic Icelandic	ISO8859-1 ISO8859-1	819 819
it it_IT	Italian - Italy Italian - Italy	ISO8859-1 ISO8859-1	819 819
it_CH	Italian - Switzerland	ISO8859-1	819
iw iw_IL	Hebrew Hebrew	ISO8859-8 ISO8859-8	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	ISO8859-5 ISO8859-5	915 915
nl nl_NL	Dutch - Netherlands Dutch - Netherlands	ISO8859-1 ISO8859-1	819 819

Table 13. Locales and CCSIDs (continued)

Locale	Language	code set	CCSID
nl_BE	Dutch - Belgium	ISO8859-1	819
no no_NO	Norwegian Norwegian	ISO8859-1 ISO8859-1	819 819
pl pl_PL	Polish Polish	ISO8859-2 ISO8859-2	912 912
POSIX	English	ISO8859-1	819
pt pt_PT	Portuguese Portuguese	ISO8859-1 ISO8859-1	819 819
ro ro_RO	Romanian Romanian	ISO8859-2 ISO8859-2	912 912
ru ru_RU ru_SU	Russian Russian Russian	ISO8859-5 ISO8859-5 ISO8859-5	915 915 915
sh sh_SP sh_YU	Serbocroatian Serbocroatian Serbocroatian	ISO8859-2 ISO8859-2 ISO8859-2	912 912 912
sl sl_SL	Slovene Slovene	ISO8859-2 ISO8859-2	912 912
sk sk_SK	Slovak Slovak	ISO8859-2 ISO8859-2	912 912
sr sr_SP	Serbian Cyrillic Serbian Cyrillic	ISO8859-5 ISO8859-5	915 915
sv sv_SE	Swedish Swedish	ISO8859-1 ISO8859-1	819 819
tr tr_TR	Turkish Turkish	ISO8859-9 ISO8859-9	920 920
zh	Simplified Chinese	eucCN	1383
zh_TW	Traditional Chinese	eucTW	964
zh_TW.BIG5	Traditional Chinese	BIG5	950

For further information about interplatform support for these locales, see the *MQSeries Application Programming Reference* book.

Data-conversion information in that book applicable to MQSeries for Sun Solaris, V5.1 applies also to MQSeries for Compaq Tru64 UNIX, V5.1.

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. Full details of the euro-supporting locales for MQSeries for Compaq Tru64 UNIX were not available at the time of print. Refer to the README file supplied with the product, and the MQSeries euro Web site at:

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

  for more information.

- 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 Compaq Tru64 UNIX 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:

BookManager	FFST	First Failure Support Technology
IBM	IBMLink	MQSeries
SupportPac		

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

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

UNIX is a registered trademark in the United States and other countries and is licensed exclusively through The Open Group.

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

Index

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

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 request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should 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:

  Information Development Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  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 you use, ensure that you include:

- The publication number and title
- 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.

GC34-5684-00

Spine information:

MQSeries® for Compaq Tru64 UNIX®

MQSeries for Compaq Tru64 UNIX, V5.1 Quick Beginnings

Version 5.1