MQSeries

Clients

GC33-1632-05

MQSeries

Clients

GC33-1632-05

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix, “Notices” on page 151.

Sixth edition (February 1998)

This edition applies to the following products:

- MQSeries for AIX Version 5
- MQSeries for AT&T GIS UNIX Version 2 Release 2
- MQSeries for Digital OpenVMS Version 2 Release 2
- MQSeries for HP-UX Version 5
- MQSeries for MVS/ESA Version 1 Release 2
- MQSeries for OS/2 Warp Version 5
- MQSeries for AS/400 Version 4 Release 2
- MQSeries for SINIX and DC/OSx Version 2 Release 2
- MQSeries for SunOS Version 2 Release 2
- MQSeries for Sun Solaris Version 5
- MQSeries for Tandem NonStop Kernel Version 2.2
- MQSeries for Windows NT Version 5

and to any subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page titled “Sending your comments to IBM”. If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories,
Information Development,
Mail Point 095,
Hursley Park,
Winchester,
Hampshire,
England,
SO21 2JN

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

Contents

Figures

Tables

About this book

This book contains information about the MQSeries client and server environment. It describes how to install an MQSeries client, how to configure the communication links and how to set up MQSeries channels so that your MQSeries applications can run on the client machine.

The MQSeries environment variables are described and there are chapters about building and running your applications on an MQSeries client.

Most of the information you need to know about MQSeries clients is in this book. Some of the reference material in the other MQSeries books includes information about MQSeries clients. That reference material is not repeated here.

This book is intended for system administrators, for anyone who installs and configures MQSeries products for the client-server environment and for application programmers who write programs to make use of the Message Queue Interface (MQI).

Terms used in this book

In this book, references to "UNIX systems" include:

AIX
AT&T** GIS UNIX
(This platform has become NCR UNIX)
HP-UX**
SINIX** and DC/OSx**
SunOS**
Sun Solaris**

References to MQSeries for "UNIX systems" include:

IBM MQSeries for AIX Version 5
IBM MQSeries for AT&T GIS UNIX Version 2.2
IBM MQSeries for HP-UX Version 5
IBM MQSeries for SINIX and DC/OSx** Version 2.2
IBM MQSeries for SunOS Version 2.2
IBM MQSeries for Sun Solaris Version 5

What you need to know

You should have:

- Experience in installing and configuring the system you use for the server: AS/400, Digital OpenVMS, MVS/ESA, OS/2, Tandem NonStop Kernel (NSK), UNIX systems, or Windows NT
- Experience with any client platforms that you will be using, for example, DOS, Windows 3.1, and Windows 95
- Understanding of the purpose of the Message Queue Interface (MQI)
- Experience of MQSeries programs in general, or familiarity with the content of the other MQSeries publications

How to use this book

Read Chapter 1, “Overview” on page 5 first as a brief introduction. There you will see “How do I set up an MQSeries client?” on page 6 which gives you a list of tasks that you may need to carry out, and this guides you through the rest of the book.

MQSeries publications

This section describes the documentation available for all current MQSeries products.

MQSeries cross-platform publications

Most of these publications, which are sometimes referred to as the MQSeries “family” books, apply to all MQSeries Level 2 products. The latest MQSeries Level 2 products are:

- MQSeries for AIX V5.0
- MQSeries for AS/400 V4R2
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Digital OpenVMS V2.2
- MQSeries for HP-UX V5.0
- MQSeries for MVS/ESA V1.2
- MQSeries for OS/2 Warp V5.0
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for SunOS V2.2
- MQSeries for Sun Solaris V5.0
- MQSeries for Tandem NonStop Kernel V2.2
- MQSeries Three Tier
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT V5.0

Any exceptions to this general rule are indicated. (Publications that support the MQSeries Level 1 products are listed in “MQSeries Level 1 product publications” on page xii. For a functional comparison of the Level 1 and Level 2 MQSeries products, see the *MQSeries Planning Guide*.)

MQSeries Brochure
The *MQSeries Brochure*, G511-1908, gives a brief introduction to the benefits of MQSeries. It is intended to support the purchasing decision, and describes some authentic customer use of MQSeries.

MQSeries: An Introduction to Messaging and Queuing
MQSeries: An Introduction to Messaging and Queuing, GC33-0805, describes briefly what MQSeries is, how it works, and how it can solve some classic interoperability problems. This book is intended for a more technical audience than the *MQSeries Brochure*.

MQSeries Planning Guide
The *MQSeries Planning Guide*, GC33-1349, 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.

MQSeries Intercommunication
The *MQSeries Intercommunication* book, SC33-1872, 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, and (3) create and configure MQSeries channels. The use of channel exits is also described.

MQSeries Clients
The *MQSeries Clients* book, GC33-1632, describes how to install, configure, use, and manage MQSeries client systems.

MQSeries System Administration
The *MQSeries System Administration* book, SC33-1873, supports day-to-day management of local and remote MQSeries objects. It includes topics such as security, recovery and restart, transactional support, problem determination, the dead-letter queue handler, and the MQSeries links for Lotus Notes**. It also includes the syntax of the MQSeries control commands.

This book applies to the following MQSeries products only:

- MQSeries for AIX V5.0
- MQSeries for HP-UX V5.0
- MQSeries for OS/2 Warp V5.0
- MQSeries for Sun Solaris V5.0
- MQSeries for Windows NT V5.0

MQSeries Command Reference
The *MQSeries Command Reference*, SC33-1369, contains the syntax of the MQSC commands, which are used by MQSeries system operators and administrators to manage MQSeries objects.

MQSeries Programmable System Management
The *MQSeries Programmable System Management* book, SC33-1482, provides both reference and guidance information for users of MQSeries events, programmable command formats (PCFs), and installable services.

MQSeries Messages
The *MQSeries Messages* book, GC33-1876, which describes “AMQ” messages issued by MQSeries, applies to these MQSeries products only:

- MQSeries for AIX V5.0
- MQSeries for HP-UX V5.0
- MQSeries for OS/2 Warp V5.0
- MQSeries for Sun Solaris V5.0
- MQSeries for Windows NT V5.0
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1

This book is available in softcopy only.

MQSeries Application Programming Guide
The *MQSeries Application Programming Guide*, SC33-0807, 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.

MQSeries Application Programming Reference
The *MQSeries Application Programming Reference*, SC33-1673, 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.

MQSeries Application Programming Reference Summary
The *MQSeries Application Programming Reference Summary*, SX33-6095, summarizes the information in the *MQSeries Application Programming Reference* manual.

MQSeries Using C++
MQSeries Using C++, SC33-1877, provides both guidance and reference information for users of the MQSeries C++ programming-language binding to the MQI. MQSeries C++ is supported by V5.0 of MQSeries for AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, and by MQSeries clients supplied with those products and installed in the following environments:

- AIX
- HP-UX
- OS/2
- Sun Solaris
- Windows NT
- Windows 3.1
- Windows 95

| MQSeries C++ is also supported by MQSeries for AS/400 V4R2.

MQSeries platform-specific publications

Each MQSeries product is documented in at least one platform-specific publication, in addition to the MQSeries family books.

MQSeries for AIX

MQSeries for AIX V5.0 Quick Beginnings, GC33-1867

| **MQSeries for AS/400**

| *MQSeries for AS/400 Version 4 Release 2 Licensed Program Specifications*,
| GC33-1958

| *MQSeries for AS/400 Version 4 Release 2 Administration Guide*, GC33-1956

| *MQSeries for AS/400 Version 4 Release 2 Application Programming Reference*
| *(RPG)*, SC33-1957

MQSeries for AT&T GIS UNIX

MQSeries for AT&T GIS UNIX Version 2.2 System Management Guide, SC33-1642

MQSeries for Digital OpenVMS

MQSeries for Digital OpenVMS Version 2.2 System Management Guide, GC33-1791

MQSeries for HP-UX

MQSeries for HP-UX V5.0 Quick Beginnings, GC33-1869

MQSeries for MVS/ESA

MQSeries for MVS/ESA Version 1 Release 2 Licensed Program Specifications, GC33-1350

MQSeries for MVS/ESA Version 1 Release 2 Program Directory

MQSeries for MVS/ESA Version 1 Release 2 System Management Guide, SC33-0806

MQSeries for MVS/ESA Version 1 Release 2 Messages and Codes, GC33-0819

MQSeries for MVS/ESA Version 1 Release 2 Problem Determination Guide, GC33-0808

MQSeries for OS/2 Warp

MQSeries for OS/2 Warp V5.0 Quick Beginnings, GC33-1868

MQSeries link for R/3

MQSeries link for R/3 Version 1.0 User’s Guide, GC33-1934

MQSeries for SINIX and DC/OSx

MQSeries for SINIX and DC/OSx Version 2.2 System Management Guide, GC33-1768

MQSeries for SunOS

MQSeries for SunOS Version 2.2 System Management Guide, GC33-1772

MQSeries for Sun Solaris

MQSeries for Sun Solaris V5.0 Quick Beginnings, GC33-1870

| **MQSeries for Tandem NonStop Kernel**

| *MQSeries for Tandem NonStop Kernel Version 2.2 System Management*
| *Guide*, GC33-1893

MQSeries Three Tier

MQSeries Three Tier Administration Guide, SC33-1451
MQSeries Three Tier Reference Summary, SX33-6098
MQSeries Three Tier Application Design, SC33-1636
MQSeries Three Tier Application Programming, SC33-1452

MQSeries for Windows

MQSeries for Windows Version 2.0 User’s Guide, GC33-1822

MQSeries for Windows Version 2.1 User’s Guide, GC33-1965

MQSeries for Windows NT

MQSeries for Windows NT V5.0 Quick Beginnings, GC33-1871

MQSeries Level 1 product publications

For information about the MQSeries Level 1 products, see the following publications:

MQSeries: Concepts and Architecture, GC33-1141

MQSeries Version 1 Products for UNIX Operating Systems Messages and Codes, SC33-1754

MQSeries for SCO UNIX Version 1.4 User's Guide, SC33-1378

MQSeries for UnixWare Version 1.4.1 User's Guide, SC33-1379

MQSeries for VSE/ESA Version 1 Release 4 Licensed Program Specifications, GC33-1483

MQSeries for VSE/ESA Version 1 Release 4 User's Guide, SC33-1142

Softcopy books

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

BookManager format

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

PostScript format

The MQSeries library is provided in PostScript (.PS) format with many MQSeries products, including all MQSeries V5.0 products. Books in PostScript format can be printed on a PostScript printer or viewed with a suitable viewer.

HTML format

The MQSeries documentation is provided in HTML format with these MQSeries products:

- MQSeries for AIX V5.0
- MQSeries for HP-UX V5.0
- MQSeries for OS/2 Warp V5.0
- MQSeries for Sun Solaris V5.0
- MQSeries for Windows NT V5.0

The MQSeries books are also available from the MQSeries product family Web site:

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

Information Presentation Facility (IPF) format

In the OS/2 environment, the MQSeries documentation is supplied in IBM IPF format on the MQSeries product CD-ROM.

Windows Help format

The *MQSeries for Windows User's Guide* is provided in Windows Help format with MQSeries for Windows Version 2.0 and MQSeries for Windows Version 2.1.

MQSeries information available on the Internet

MQSeries web site

The MQSeries product family Web site is at:

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

By following links from this Web site you can:

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

Related publications

MQSeries Client for VM/ESA
VM/ESA VMS Application Development Guide, Version 2, Release 3.0 , SC24-5761

Related publications

Summary of changes

| Changes to the previous edition are marked in the left-hand margin with bars.

| Changes for this edition

| The following products are included:

| • MQSeries for AS/400 Version 4 Release 2
| • MQSeries for Tandem NonStop Kernel Version 2.2
| • MQSeries Client for VM/ESA Version 2 Release 3

Changes for the Fifth Edition

The MQSeries Version 5 products were included:

- MQSeries for AIX Version 5
- MQSeries for HP-UX Version 5
- MQSeries for OS/2 Warp Version 5
- MQSeries for Sun Solaris Version 5
- MQSeries for Windows NT Version 5

The following products were also included:

- MQSeries for Digital OpenVMS Version 2.2
- MQSeries for MVS/ESA Version 1 Release 2
- MQSeries for OS/400 Version 3 Release 7

Changes for the Fourth Edition

The following products were included:

- MQSeries for MVS/ESA Version 1 Release 1.4
- MQSeries for OS/400 Version 3 Release 2
- MQSeries for SINIX and DC/OSx Version 2.2
- MQSeries for SunOS Version 2.2
- MQSeries for Sun Solaris Version 2.2

Part 1. Overview and installation

Chapter 1. Overview

What is an *MQSeries client*? What are the benefits of using MQSeries clients, and how do I set up a client and server system? These questions are answered here.

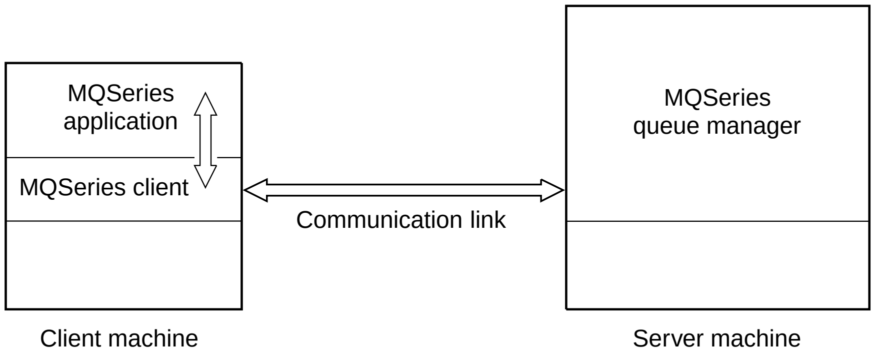
What is an MQSeries client?

An MQSeries client is part of the MQSeries product that can be installed on its own, on a separate machine from the Base product and Server. You can run an MQSeries application on an MQSeries client and it can interact, by means of a communications protocol, with one or more MQSeries servers and connect to their queue managers.

These are the platforms that can be used. The combinations depend on which MQSeries product you are using; see "Support for MQSeries clients" on page 9. Other MQSeries clients are also available; see "MQSeries clients on other platforms" on page 11.

MQSeries client
- Digital OpenVMS
- OS/2
- UNIX systems
- VM/ESA
- Windows NT
- Windows 3.1
- Windows 95
- DOS

MQSeries server
- Digital OpenVMS
- OS/2
- UNIX systems
- Windows NT
- AS/400
- MVS/ESA
- Tandem NSK



An application that you want to run in the MQSeries client environment must first be linked with the relevant client library. When the application issues an MQI call, the MQSeries client code directs the request to a queue manager, where it is processed and from where a reply is sent back to the MQSeries client.

The link between the application and the MQSeries client code is established dynamically at runtime, except in the case of DOS, when it is a static link.

Why use MQSeries clients?

Using MQSeries clients is an efficient way of implementing MQSeries messaging and queuing.

You can have an application that uses the MQI running on one machine and the queue manager running on a different machine, either physical or virtual. The benefits of doing this are:

- There is no need for a full MQSeries implementation on the client machine; for example, it could be a DOS, a Windows 3.1, or a Windows 95 platform.
- Hardware requirements on the client system are reduced.
- System administration requirements are reduced.
- An MQSeries application running on a client can connect to multiple queue managers on different systems.
- Alternative channels using different transmission protocols may be used.

What applications run on an MQSeries client?

The full MQI is supported in the client environment and this enables almost any MQSeries application to be relinked to run on an MQSeries client. Link the application on the MQSeries client to the MQIC library, rather than to the MQI library. The exceptions are:

- An application that needs syncpoint coordination with other resource managers.
- Get(signal) on MVS/ESA is not supported.

Note: An application running on an MQSeries client may connect to more than one queue manager concurrently, or use a queue manager name with an asterisk (*) on an MQCONN or MQCONNX call (see Examples in Chapter 12, “Running applications on MQSeries clients” on page 135). The application will have to be changed if you want to link to the queue manager libraries instead of the client libraries, as this function will not be available.

How do I set up an MQSeries client?

This book tells you how to set up and use an MQSeries client. You need to have an MQSeries server already installed and working on another machine, for your client to connect to. These are the tasks you need to carry out:

1. Check that you have a suitable platform for an MQSeries client and that the hardware and software satisfy the requirements. See Chapter 2, “Preparing for installation” on page 9 for information.
2. If your communication links are configured and connected, go on to the next step, otherwise go to Chapter 6, “Configuring communication links” on page 75.
3. Decide how you are going to install MQSeries on your client machine, and then follow the instructions for your particular combination of client and server platforms. See either Chapter 3, “Installing MQSeries client components from Version 5 products” on page 31 or Chapter 4, “Installing MQSeries clients with non-Version 5 products” on page 51.

4. Make sure that your installation is working. See Chapter 5, “Verifying the installation” on page 65.

5. Now you have a verified MQSeries client installation, consider whether you need to take any action on security. See Chapter 9, “Setting up MQSeries client security” on page 117 for details.

6. Next set up the *channels* between MQSeries client and server that are required by the MQSeries applications you want to run on the client. See Chapter 7, “Using channels” on page 99 for the detailed steps to take. You may need to use MQSeries *environment variables* to set up the channels. See Chapter 8, “Using MQSeries environment variables” on page 111 for details.

7. MQSeries applications are fully described in the *MQSeries Application Programming Guide*. There are some differences to consider when designing, building and running applications in the MQSeries client environment. For details see:

   Chapter 10, “Using the message queue interface (MQI)” on page 123
   Chapter 11, “Building applications for MQSeries clients” on page 127
   Chapter 12, “Running applications on MQSeries clients” on page 135
   Chapter 13, “Solving problems” on page 141

Chapter 2. Preparing for installation

This chapter details the platform support for clients, explains the communications protocols used, and shows how MQSeries clients fit into your network. The hardware and software requirements for each client platform are listed.

Hardware and software requirements

You can find hardware and software requirements for the client platforms as follows:

Desktop clients

- DOS - page 17
- OS/2 Warp - page 20
- Windows 3.1 - page 27
- Windows 95 - page 28
- Windows NT - page 29

UNIX clients

- AIX - page 14
- AT&T GIS UNIX (NCR UNIX) - page 16
- HP-UX - page 19
- SINIX and DC/OSx - page 22
- SunOS - page 24
- Sun Solaris - page 25

Other clients

- Digital OpenVMS - page 18
- VM/ESA - page 26

For your server platform hardware and software requirements, see the *Quick Beginnings* book for your platform (MQSeries Version 5 products), the *MQSeries for MVS/ESA Program Directory*, the *MQSeries for AS/400 Administration Guide* or the *System Management Guide* (other products).

For capacity planning information, see the *MQSeries Planning Guide*.

Support for MQSeries clients

The platform support for MQSeries clients and servers is as follows. Any of the MQSeries products listed is installed as a *Base product and Server* (*Base product and Distributed Queuing without CICS feature, and Client Attachment feature* on MQSeries for MVS/ESA). These MQSeries products can accept connections from the MQSeries clients on the platforms listed, subject to differences in coded character set identifier (CCSID) and communications protocol.

Note: If you are using previous versions of MQSeries products, make sure that code conversion from the CCSID of your client is supported by the server. See the Language support tables in the *MQSeries Application Programming Reference*.

The MQSeries products covered by this book:

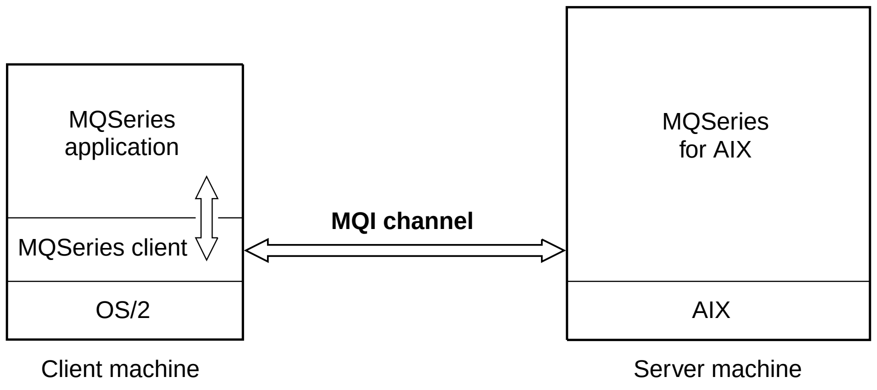
- Version 5 products:
  - MQSeries for AIX Version 5
  - MQSeries for HP-UX Version 5
  - MQSeries for OS/2 Warp Version 5
  - MQSeries for Sun Solaris Version 5
  - MQSeries for Windows NT Version 5
- Non-Version 5 products:
  - MQSeries for AS/400 V4R2
  - MQSeries for AT&T GIS UNIX Version 2.2
  - MQSeries for Digital OpenVMS Version 2.2
  - MQSeries for MVS/ESA Version 1 Release 2
  - MQSeries for SINIX and DC/OSx Version 2.2
  - MQSeries for SunOS Version 2.2
  - MQSeries for Tandem NonStop Kernel Version 2.2

can accept connections from MQSeries clients on:

- AIX
- AT&T GIS UNIX
  (this platform has become NCR UNIX)
- Digital OpenVMS
- DOS
- HP-UX
- OS/2
- SINIX and DC/OSx
- SunOS
- Sun Solaris
- VM/ESA
- Windows 3.1
- Windows 95
- Windows NT

Note: MQSeries for Windows Version 2.0 and MQSeries for Windows Version 2.1 are ***not*** included in this book. The MQSeries for Windows products are the MQSeries queue managers for the Microsoft Windows platforms. They are designed to minimize system requirements for workstations with relatively modest specifications. You cannot use an MQSeries for Windows queue manager as an MQSeries client, nor can you use it to support its own MQSeries clients. For more information see the *MQSeries for Windows Version 2.0 User's Guide* and the *MQSeries for Windows Version 2.1 User's Guide*.

Here is an example of an MQSeries client and server system:



The MQI is available to applications running on the client platform; the queues and other MQSeries objects are held on a queue manager that you have installed on a server machine (for an explanation of *MQI channel* see "What is a channel?" on page 99).

MQSeries clients on other platforms

The MQSeries clients included in this book are the ones supplied with the MQSeries products listed in "Support for MQSeries clients" on page 9. Each
| MQSeries product (except MQSeries for AS/400, MQSeries for Tandem NonStop
| Kernel, and MQSeries for MVS/ESA) supplies files for clients on the same platform as the server and for a number of other platforms. For details see "MQSeries for Digital OpenVMS, and UNIX platforms" on page 53 and "Clients supplied with MQSeries Version 5 products" on page 31.

Further MQSeries Clients are available through the Internet as Supportacs. See "MQSeries information available on the Internet" on page xiii.

How the client connects to the server

An application running in the MQSeries client environment runs in synchronous mode, as there must be an active connection between the client and server machines.

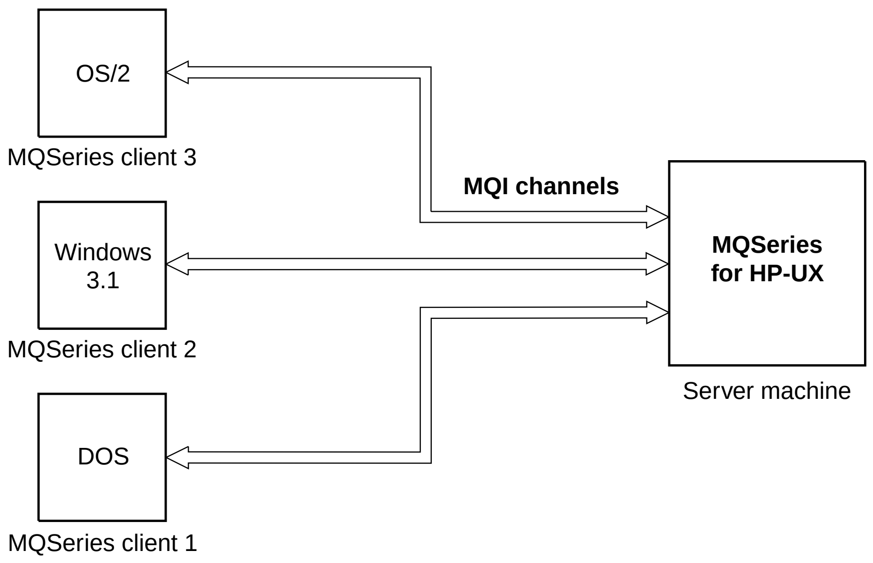
The connection is made by an application issuing an MQCONN or MQCONNX call. When the call succeeds, the MQI channel remains connected until the application issues a MQDISC call. This is the case for every queue manager that an application needs to connect to.

Client and queue manager on the same machine

You can also run an application in the MQSeries client environment where your machine has a queue manager installed. In this situation you have the choice of linking to the queue manager libraries or the client libraries, but remember that if you link to the client libraries, you still need to define the channel connections. This can be useful during the development phase of an application. You can test your code on your own machine, with no dependency on others, and be confident that it will still work when you move it to a full MQSeries environment.

Clients on different platforms

Here is another example of an MQSeries client and server system. In this example the server machine communicates with three MQSeries clients on different platforms.



Other more complex environments are possible. An MQSeries client can connect to more than one queue manager, for example.

Communications

MQSeries clients use MQI channels to communicate with the server. A channel definition must be created at both the MQSeries client and server ends of the connection. How to do this is explained in “Connecting the MQSeries client and server - channel definitions” on page 100.

The transmission protocols possible are shown in the following table:

Table 1. Transmission protocols for MQI channels

Client platform	**LU 6.2**	**TCP/IP**	**NetBIOS**	**SPX**	**DECnet**
Digital OpenVMS	√	√			√
DOS		√	√	√	
OS/2	√	√	√	√	
UNIX platforms	√	√			
| VM/ESA	√	√			
Windows 3.1		√	√	√	
Windows 95		√	√	√	
Windows NT	√	√	√	√	

See also Table 2 on page 76 for the possible combinations of MQSeries client and server platforms, using these transmission protocols.

An MQSeries application on an MQSeries client can use all the MQI calls in the same way as when the queue manager is local. MQCONN or MQCONNX associates the MQSeries application with the selected queue manager, creating a *connection handle*. Other calls using that connection handle are then processed by the connected queue manager. This MQSeries client communication is synchronous, in contrast to communication between queue managers, which is connection- and time-independent.

The transmission protocol is specified via the channel definition and does not affect the application. For example, the same Windows 3.1 application can connect to one queue manager over TCP/IP and to another queue manager over NetBIOS.

Performance considerations

The transmission protocol you use may affect the performance of the MQSeries client and server system.

For dial-up support over a slow phone line it may be advisable to use channel exits to compress the data transmitted.

AIX client: hardware and software required

An MQSeries client can run on an IBM RISC System/6000, capable of running AIX V4.1.4. Any other trademarked AIX system may be used, whether from IBM or other vendors such as Bull, Zenith, or Motorola. There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the MQSeries client code, the access methods, and the application programs.

Programming requirements

The following are prerequisites for MQSeries applications running on an AIX client.

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

For TCP/IP connectivity:

- TCP/IP (in the operating system)

For SNA connectivity:

- IBM Communications Server for AIX, V4.0 (5765-652)

Workstation MQSeries clients

MQSeries client code for AIX workstations is distributed with the server code for all MQSeries Version 5 products. Later levels of some listed products may be required for AIX V4.2 and/or SP. Later levels of the operating system may be required to support corequisite products. The MQSeries client code operates under:

- AIX V4.1.4 (5765-393 or 5765-C34) or AIX Version 4.2 (5765-655 or 5765-C34) or later.

  Note: For V4.1, PTF U449790 is required if user data conversion of Greek, Cyrillic, Eastern European, Turkish, Japanese, or Korean language text longer than 2000 bytes is required.

  For V4.2 level 4.2.1 should be used for the same languages.

Options, not prerequisites

- IBM Directory and Security Server for AIX (5765-639), V4 and later compatible versions. This must be the U.S. Domestic version supporting DES encryption if the user wishes to run the MQSeries-supplied DCE send, receive, or message exits.

  MQSeries DCE names and security modules are provided as part of the MQSeries for AIX product.

Compilers for MQSeries applications on AIX clients

The following C compilers are supported:

- IBM C for AIX V3.1.4 (5765-423)
- IBM C Set++ for AIX V3.1 (5765-421)

The following COBOL compilers are supported:

- IBM COBOL Set for AIX V1.0 (5765-548 28H2176/33H4408)
- Micro Focus** COBOL for UNIX Version 3.1 and v4.0

The following C++ compilers are supported:

- IBM C Set++ for AIX V3.1 (5765-421)

The following PL/I compilers are supported:

- IBM PL/I set for AIX V1.1

AT&T GIS UNIX (NCR UNIX) client: hardware and software required

An MQSeries client can run on the following:

- Any AT&T GIS 34XX, 35XX, or 36XX system with minimum system disk space of 20 MB
- Any LAN adapter
- Any communications hardware supporting SNA/LU 6.2 and/or TCP/IP

Programming requirements

The following are prerequisites for MQSeries applications running on an AT&T GIS UNIX client.

Minimum supported levels are shown. Later levels, if any, will be supported unless otherwise stated.

Workstation MQSeries clients

MQSeries client code for AT&T GIS UNIX workstations is distributed with the server code for all MQSeries Version 5 products. It operates under:

- AT&T GIS UNIX SVR4 MP-RAS Version 3.0, including TCP/IP
  (this platform has become NCR UNIX SVR4 MP-RAS, R3.0)
- Appropriate LAN software, for example NFS to match TCP/IP

  (if you who plan to use NFS, please contact your service representative to obtain all available patches)
- AT&T GIS SNA Services Version 2.06 or later

Compiler for MQSeries applications on AT&T GIS UNIX clients

The following C compiler is supported:

- AT&T GIS High Performance C V1.0b

DOS client: hardware and software required

An MQSeries client can run on DOS, on a personal computer. There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the MQSeries client code, the access methods, and the application programs.

Programming requirements

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

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

- NetBIOS
- IBM TCP/IP V2.1.1 for DOS (87G7184 5621-219 (EMEA))

Workstation clients

MQSeries client code for DOS workstations is distributed with the server code for
| all MQSeries products (except MQSeries for AS/400, MQSeries for MVS/ESA, and
| MQSeries for Tandem NonStop Kernel). It operates under:

- DOS 5.0

Options, not prerequisites

- IBM TCP/IP for DOS: NetBIOS, V2.1.1 (87G7186 5622-048 (EMEA))
- IBM TCP/IP for OS/2 V3.0 is part of the base operating system
- Internet Connection Corporate kit for Windows 3.1 and Windows 95, V5.0
- Novell Netware Client for DOS/Win, V1.20
- Novell Netware Client for OS/2, V2.10
- Novell LAN Workplace, V5.1
- FTP TCP for DOS

The DOS access kit allows clients access to TCP/IP via programs that run in a DOS window under OS/2.

The Novell Netware Client for OS/2 allows clients access to SPX via programs that run in a DOS window under OS/2.

Compilers for MQSeries applications on DOS clients

The following C compilers are supported:

- Microsoft C V7.0
- Microsoft Visual C++ V1.5

Digital OpenVMS client: hardware and software required

An MQSeries client can run on Digital OpenVMS on Digital VAX or AXP (Alpha) systems with minimum system disk space of 700 blks (=350 KB) and minimum memory of 8 MB.

Connectivity

Network protocols supported are SNA LU 6.2, TCP/IP, and DECnet.

- Digital SNA Domain Gateway for Synchronous or Channel Transport
- Digital SNA Peer Server
- Any communications hardware supporting TCP/IP or DECnet

Programming requirements

The following are prerequisites for MQSeries applications running on a Digital OpenVMS client.

Minimum supported levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

- Digital SNA APPC LU6.2 Programming Interface v2.3
- Digital DECnet SNA Gateway software v1.2A
- Process Software TCPWare v5.2-3
- VAX/AXP: DECnet SNA APPC/LU 6.2 Version 2.2
- VAX/AXP: CISCO (formerly TGV) MultiNet Version 3.5 for OpenVMS
- AXP: TCP/IP Services for OpenVMS AXP Version 4.0
- Digital TCP/IP services for OpenVMS (UCX) V4.1
- VAX: TCP/IP Services for OpenVMS VAX Version 3.3
- Attachmate Pathway for OpenVMS Version 2.5.1

Workstation MQSeries Clients

MQSeries client code for Digital OpenVMS workstations is distributed with the server code for all MQSeries Version 5 products. It operates under:

- OpenVMS operating system Version 6.2

Options, not prerequisites

- DCE
  - Distributed Computing Environment for OpenVMS Version 1.3b

Compilers for MQSeries applications on Digital OpenVMS clients

The following COBOL compilers are supported:

- VAX: VAX COBOL Version 5.3
- AXP: DEC COBOL Version 2.2

The following C compilers are supported:

- AXP/VAX: DEC C Version 5.2
- AXP: DEC C++ Version 5.2
- VAX: DEC C++ Version 5.0

HP-UX client: hardware and software required

An MQSeries client can run on HP-UX on any HP 9000 Series 700 or Series 800 with minimum system disk space of 20 MB.

Programming requirements

The following are prerequisites for MQSeries applications running an HP-UX client.

Minimum supported levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

- TCP/IP (in the operating system)
- HP SNAplus2

Workstation MQSeries clients

MQSeries client code for HP-UX workstations is distributed with the server code for all MQSeries Version 5 products. It operates under:

- HP-UX Version 10.10, or later Version 10.

Options, not prerequisites

- The HP DCE/9000 version appropriate for the level of the HP-UX operating system in use, providing this is compatible with DCE Version 1.4.1 as supplied for HP-UX 10.10. This must be the U.S. Domestic version supporting DES encryption if the user wishes to run the MQSeries-supplied DCE send, receive, or message exits.

  Note: On HP-UX 10.10, it is critical to apply HP service otherwise the HP DCE product will not work. Patches are listed in the Release Note delivered with the HP DCE software; however you are recommended to contact your local HP support center to obtain an up-to-date list of the required patches.

  MQSeries DCE names and security modules are provided as part of the MQSeries for HP-UX product.

Compilers for MQSeries applications on HP-UX clients

The following COBOL compilers are supported:

- Micro Focus COBOL for UNIX Version 3.1
- COBOL Softbench Version 4.0

The following C compilers are supported:

- The compiler supplied with the operating system
- HP-UX ANSI compiler
- C Softbench Version 5.0
- HP C++ Version 3.1

The following C++ compilers are supported:

- HP C++ Version 3.1

OS/2 Warp client: hardware and software required

An MQSeries client can run on OS/2 Warp, on a personal computer. There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the MQSeries client code, the access methods, and the application programs.

Programming requirements

The following are prerequisites for MQSeries applications running an OS/2 client.

This is the minimum supported software level. Later levels, if any, will be supported unless otherwise stated.

Workstation MQSeries clients

MQSeries client code for OS/2 Warp workstations is distributed with the server code for all MQSeries Version 5 products. It operates under:

- OS/2 Warp V4.0 (84H1426)

Options, not prerequisites

- Communications Manager/2 V1.11 for OS/2 (this includes LU 6.2 and NetBIOS) (79G0257/79G0258)
- IBM Communications Server for OS/2 V4.0 (84H1802)
- Novell Netware Client for OS/2, V1.20 (for direct IPX/SPX support)
- IBM Directory and Security Server for OS/2 Warp Version 4 or later compatible versions. This must be the U.S. Domestic version supporting DES encryption if the user wishes to run the MQSeries-supplied DCE send, receive, or message exits.
- If used as a DCE server this software is known to run adequately in the following environment:
  - On a Pentium processor running 90 MHz or faster.
  - On a machine with 64 MB or more of memory.
  - Using OS/2 Warp Server V4.0 or later.
- MQSeries DCE names and security modules are provided as part of the MQSeries for OS/2 Warp product.

Compilers for MQSeries applications on OS/2 clients

The following COBOL compilers are supported:

- IBM VisualAge COBOL for OS/2 V1.1 (28H2177, 5622-793 (EMEA))
- Micro Focus COBOL Compiler Version 3.0.54 and V4.0 (32 bit)

The following C compilers are supported:

- IBM C Set++ for OS/2 V2.1 (82G3732/82G3735, 5604-535 EMEA))
- IBM VisualAge C++ for OS/2 Version 3.0
- Borland C++ Compiler Version 2.0

The following C++ compilers are supported:

- IBM VisualAge C++ for OS/2 Version 3.0

The following PL/I compilers are supported:

- IBM PL/I for OS/2 Version 1.2
- IBM VisualAge for PL/I for OS/2

SINIX and DC/OSx client: hardware and software required

An MQSeries client can run on:

- SINIX: RM200, RM300, RM400, RM600 systems with minimum system disk space of 30 MB. If DynaText books are installed, a minimum of 50 MB of system disk space is needed.
- DC/OSx: MIServer, Nile systems with minimum system disk space of 30 MB.
- Any communications hardware supporting SNA/LU 6.2 or TCP/IP.

Programming requirements

The following are prerequisites for MQSeries applications running on a SINIX and DC/OSx client.

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

- SINIX operating system SINIX-N Version 5.42C10 (for RM200, RM300, RM400) or SINIX-Y Version 5.42A40 (for RM600)
- DC/OSx operating system Version 1.1-cd079 or later

MQSeries Clients

Client code for SINIX and DC/OSx workstations is distributed with the server code for all MQSeries Version 5 products.

Connectivity

- SINIX: SNA

  TRANSIT-SERVER 3.4 (SNA Communication Server Version)
  TRANSIT-CLIENT 3.4 (SNA Comm. Client / Local Functions)
  TRANSIT-CPIC 3.4 (SNA LU 6.2 Communication and CPI-C)
- SINIX: OpenNet TCP/IP
- DC/OSx: TCP/IP Version 1.0
- DC/OSx: SNA requires LU 6.2 SW version 1.3 and
  - To support the ISC-2 (Intelligent Synchronous Controller) serial line:
    - Comm Services V 1.2
    - ISC with SNA engine V 1.3
  - To support the ILC-T (Intelligent LAN Controller, Token ring) interface:
    - Comm Services V 1.2
    - Token Ring Mac interface V 1.3
  - To support the SNA on the ESCON IBM Channel link:
    - XVI/ESCON Driver 1.0
- DCE
  - SINIX: Version 1.03A00 or later

Compilers for MQSeries applications on SINIX and DC/OSx clients

The following COBOL compilers are supported:

- SINIX: Micro Focus COBOL version 3.2
- DC/OSx: Micro Focus COBOL version 3.2

The following C compilers are supported:

- SINIX: C compiler (C-DS, MIPS) version 1.1
- DC/OSx: C4.0 compiler version 4.0.1

SunOS client: hardware and software required

An MQSeries client can run only on the Sun SPARC hardware with a minimum system disk space of 25 MB

Connectivity

- Any communications hardware supporting SNA/LU 6.2 and/or TCP/IP

Programming requirements

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

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

- SunOS UNIX Version 4.1.3 or later 4.1.x, to include TCP/IP

MQSeries Clients

Client code for SunOS workstations is distributed with the server code for all MQSeries Version 5 products.

Connectivity

- SunLink SNA Peer-to-Peer Version 7.0
- TCP/IP as part of the base operating system

Compilers for MQSeries applications on SunOS clients

The following COBOL compiler is supported:

- Micro Focus COBOL Version 3.0

The following C compiler is supported:

- SparcCompiler C Version 3.0.1

Sun Solaris client: hardware and software required

An MQSeries client can run only on:

- Sun SPARC desktop or server
- Sun UltraSPARC desktop or server

with a minimum system disk space of 25 MB. An additional 25 MB of disk space is required if DynaText is to be installed.

Connectivity

- Any communications hardware supporting SNA/LU 6.2 and/or TCP/IP

Programming requirements

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

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

- Sun Solaris V2.5.1 or later 2.X to include TCP/IP

  It is recommended that the latest levels of all relevant patches to the base operating system are installed.

MQSeries Clients

Client code for Sun Solaris workstations is distributed with the server code for all MQSeries Version 5 products.

Connectivity

- SunLink SNA Peer-to-Peer Version 9.0 or later V9.X
- TCP/IP as part of the base operating system
- If token ring is to be used: SunLink Token Ring Interface /SBus, V3.0.2. This requires patch 102463

Compilers for MQSeries applications on Sun Solaris clients

The following COBOL compiler is supported:

- Micro Focus COBOL for UNIX Version 3.2

The following C compiler is supported:

- SPARCompiler C Version 4.0

The following C++ compiler is supported:

- SPARCompiler C++ Version 4.1

VM/ESA client: hardware and software required

An MQSeries client can run on any CMS system that supports the programming prerequisites below.

Programming requirements

The following are prerequisites for MQSeries applications running on a VM/ESA client.

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

- TCP/IP Release 2.4
- VTAM LU 6.2

MQSeries clients

MQSeries client code for VM/ESA is distributed with the VM/ESA product. It operates under:

- VM/ESA Version 2 Release 3
- LE/370 Release 1.6

Compilers for MQSeries applications on VM/ESA clients

The following Assembler language compiler supported:

- IBM Assembler

The following COBOL compiler is supported:

- IBM VS COBOL II

The following C compiler is supported:

- IBM C for VM Release 3.1

The following PL/I compiler is supported:

- IBM OS/PL/I Release 2.3

The following Rexx compiler is supported:

- IBM VM/ESA Rexx/VM

Windows 3.1 client: hardware and software required

An MQSeries client can run on Windows 3.1, on a personal computer. There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the MQSeries client code, the access methods, and the application programs.

Programming requirements

The following are prerequisites for MQSeries applications running on a Windows 3.1 client.

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

- NetBIOS
- IBM TCP/IP V2.1.1 for DOS
- SPX

Workstation clients

MQSeries client code for Windows 3.1 workstations is distributed with the server code for all MQSeries products (except MQSeries for AS/400, MQSeries for MVS/ESA, and MQSeries for Tandem NonStop Kernel). It operates under:

- Windows 3.1
- WIN-OS/2 environment under OS/2
- Windows 95 in 16-bit mode
- Windows for Workgroups

Options, not prerequisites

- TCP/IP for OS/2 V2.0. The base kit is required. The DOS access kit allows clients access to TCP/IP via programs that run under WIN-OS/2.
- TCP/IP V2.1.1 for DOS/Windows
- Novell Netware client for DOS/Win31 V1.20
- Novell Netware client for OS/2 V2.1 (allows clients to access SPX via programs that run under WIN-OS/2)

Compilers for MQSeries applications on Windows 3.1 clients

The following COBOL compilers are supported:

- Micro Focus COBOL for Windows Version 3.3

The following C compilers are supported:

- Microsoft C/C++ Version 7.0
- Microsoft Visual C++ for Windows Version 2.0

The following C++ compilers are supported:

- Microsoft Visual C++ Version 1.5

Windows 95 client: hardware and software required

An MQSeries client can run on Windows 95 on a personal computer. There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the MQSeries client code, the access methods, and the application programs.

Programming requirements

The following are prerequisites for MQSeries applications running on a Windows 95 client.

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

- TCP/IP (in the operating system)
- SPX (in the operating system)
- NetBIOS (in the operating system)

Workstation MQSeries clients

MQSeries client code for Windows 95 workstations is distributed with the server code for all MQSeries Version 5 products. It operates under:

- Windows 95

Options, not prerequisites

- IBM DCE for Windows 95: V1.1 and later compatible versions. (This will become IBM Directory and Security Server for Windows 95 in 1998).

  | You cannot use the supplied DCE security exit (described in the book
  | *MQSeries Intercommunication*) from an MQSeries client on Windows 95
  | connected to an MQSeries for HP-UX server or an MQSeries for Sun Solaris
  | server.

Compilers for MQSeries applications on Windows 95 clients

The following COBOL compilers are supported:

- Micro Focus COBOL Workbench Version 4.0

The following C compilers are supported:

- IBM VisualAge C++ for Windows Version 3.5 (33H4979)
- Microsoft Visual C++ for Windows 95/NT Version 4.0

The following C++ compilers are supported:

- IBM VisualAge C++ for Windows Version 3.5 (33H4979)
- Microsoft Visual C++ for Windows 95/NT Version 4.0 and Version 5.0

Windows NT client: hardware and software required

An MQSeries client can run on Windows NT on any Intel** 486 processor based IBM PC machine (or equivalent). There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the MQSeries client code, the access methods, and the application programs.

Programming requirements

The following are prerequisites for MQSeries applications running on a Windows NT client.

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.

Connectivity

- IBM Communications Server for Windows NT, V5.0 (4231747)
- IBM Personal Communications for Windows NT, V4.1
- Attachmate Extra! Personal Client, V6.1 & V6.2
- Microsoft SNA Server, V2.11 or V3
- TCP/IP, NetBIOS, and SPX are part of the base operating system
- OnNet SDK for Windows
- FTP Software

Workstation MQSeries clients

MQSeries client code for Windows NT workstations is distributed with the server code for all MQSeries Version 5 products. It operates under:

- Microsoft Windows NT V3.5.1 with service pack 5 applied to it, or Windows NT V4, to include TCP/IP, NetBIOS, and SPX.

Compilers for MQSeries applications on Windows NT clients

The following COBOL compilers are supported:

- Micro Focus COBOL for Windows NT Version 3.1J
- Micro Focus COBOL for Windows NT Version 3.3

The following C compilers are supported:

- IBM VisualAge C++ Version 3.5 (33H4979)
- Microsoft Visual C++ for Windows 95/NT Version 4.0 and Version 5.0

The following C++ compilers are supported:

- IBM VisualAge C++ Version 3.5 (33H4979)
- Microsoft Visual C++ Version 4.0

The following PL/I compilers are supported:

- IBM PL/I for Windows V1.2,
- IBM VisualAge for PL/I for Windows

Chapter 3. Installing MQSeries client components from Version 5 products

The MQSeries Version 5 products are:

- MQSeries for AIX Version 5
- MQSeries for HP-UX Version 5
- MQSeries for OS/2 Warp Version 5
- MQSeries for Sun Solaris Version 5
- MQSeries for Windows NT Version 5

MQSeries Version 5 products include an easy installation feature that helps you install MQSeries clients quickly. If you are using another MQSeries product, see Chapter 4, "Installing MQSeries clients with non-Version 5 products" on page 51.

Installing an MQSeries client and server system

You install the MQSeries client and server system from the two CD-ROMs supplied:

1. Install the MQSeries server on your server machine using the MQSeries Server CD-ROM, as detailed in the *Quick Beginnings* book for your platform.
2. Install the MQSeries client components on your client machine, or on several client machines, using the MQSeries Client CD-ROM, as explained in this chapter.

See the step by step instructions for your client platform:

- "Installing on AIX" on page 36
- "Installing on DOS" on page 39
- "Installing on HP-UX" on page 39
- "Installing on OS/2 Warp" on page 41
- "Installing on Sun Solaris" on page 43
- "Installing on Windows 3.1" on page 45
- "Installing on Windows NT or Windows 95" on page 47

Clients supplied with MQSeries Version 5 products

Each MQSeries Version 5 product supplies software, including the easy installation feature, for clients on the following platforms:

- AIX
- DOS
- HP-UX
- OS/2
- Sun Solaris
- Windows 3.1
- Windows 95
- Windows NT

Further MQSeries clients are available through the Internet, as described in "MQSeries clients from IBM Transaction Processing SupportPacs" on page 51. and "MQSeries information available on the Internet" on page xiii.

Components you can install

During the installation you will be given a choice of components that you can install on your non-server machine. For details, see the list for your platform:

- "Components for UNIX platforms"
- "Components for OS/2 Warp and Windows NT" on page 33
- "Components for Windows 95" on page 34
- "Components for DOS and Windows 3.1" on page 34

Components for UNIX platforms

The components you can install on AIX, HP-UX, and Sun Solaris systems are:

MQSeries Client
The MQSeries client code for your UNIX platform.

Samples
Sample application programs.

Support for DCE in Samples
The DCE samples support. This should be installed if, and only if, you are going to use DCE.

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 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:

AIX	Java version 1.1.1
HP-UX	Java version 1.1.2
Sun Solaris	Java Version 1.1.1

For information about Java runtime see "MQSeries information available on the Internet" on page xiii.

On Sun Solaris, you require version 2.5.1 of the Solaris Operating System plus the following patches: 103566-08, 103600-13, 103640-08.

On HP-UX, you require HP-UX Version 10.20.

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

MQSeries Internet/ Java documentation
MQSeries Java Client and Internet Gateway documentation. This is supplied in HTML format.

MQSeries Internet Gateway
| Provides access to MQSeries applications via HTML and CGI (only CGI on
| the Sun Solaris platform). The MQSeries Internet Gateway does not support
| NSAPI on the HP-UX platform.

Now go to “Installing on AIX” on page 36, “Installing on HP-UX” on page 39, or “Installing on Sun Solaris” on page 43.

Components for OS/2 Warp and Windows NT

The components you can install on OS/2 Warp and Windows NT systems are:

MQSeries Client
The MQSeries client code for your platform.

MQSeries Toolkit
This includes:

- Sample programs
- Header files that you can use when writing applications to run on the client

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 1.1.1 (or later compatible version) runtime code on your machine. On
| Windows NT, you require Microsoft Windows NT Version 4. On OS/2 the
| MQSeries client for Java must be installed on an HPFS formatted drive. For information about Java runtime see “MQSeries information available on the Internet” on page xiii.

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

MQSeries Internet/ Java documentation
MQSeries Java Client and Internet Gateway documentation. This is supplied in HTML format.

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

Now go to “Installing on OS/2 Warp” on page 41, or “Installing on Windows NT or Windows 95” on page 47.

Components for Windows 95

The components you can install on a Windows 95 system are:

MQSeries Client
The MQSeries client code for Windows 95.

MQSeries Toolkit
This includes:

- Sample programs
- Header files that you can use when writing applications to run on the client

Now go to "Installing on Windows NT or Windows 95" on page 47.

Components for DOS and Windows 3.1

The components you can install on DOS and Windows 3.1 systems are:

MQSeries Client
The MQSeries client code for your platform.

MQSeries Toolkit
This includes:

- Sample programs - some of these are required for verifying the installation of the MQSeries client/server system
- Header files that you can use when writing applications to run on the client

Now go to "Installing on DOS" on page 39, or "Installing on Windows 3.1" on page 45.

Installing MQSeries clients on the same machine as the server

To install MQSeries client components on the server machine use the MQSeries Server CD-ROM. Choose the client install option to install client components on the server machine. Do not use the MQSeries Client CD-ROM.

Remember that you still need to define the MQI *channels* between the client and the server, even when the two reside on the same machine. See Chapter 7, “Using channels” on page 99 for details.

You may install client components from the MQSeries Client CD-ROM on a machine and later want to install the MQSeries server on the same machine. If so, you must first remove all the MQSeries client components from the machine. Then use the MQSeries Server CD-ROM to install the server and the client. You cannot install the server on a machine that already has client components installed from the MQSeries Client CD-ROM.

Removing MQSeries clients

If you want to remove the MQSeries client files from your system, use the process provided to do this efficiently. Details for each platform are given in the relevant section of this chapter:

Removing an MQSeries client from:

- AIX (page 39)
- DOS (page 39)
- HP-UX (page 41)
- OS/2 (page 43)
- Sun Solaris (page 45)
- Windows NT, and Windows 95 (page 49)
- Windows 3.1 (page 46)

Installing on AIX

To install an MQSeries client on an AIX system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Note: If you plan to install an MQSeries client and server on the same machine, see "Installing MQSeries clients on the same machine as the server" on page 35.
| If you have a previous version of the MQSeries client installed on your AIX system,
| you must uninstall the old version before installing the Version 5 MQSeries client.

Before installation

Before you can install the MQSeries client on your AIX system, you:

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

  If you create separate partitions, the following directories ***must*** be on a local file system:

  – `/var/mqm`
  – `/var/mqm/log`
  – `/var/mqm/objects`

  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.

Creating the mqm user ID and group

You can create the new IDs using the System Management Interface Tool (SMIT), for which you require root authority. The procedure for this, if you use the SMIT windows, is:

1. Create the `mqm` group. You can display the required window using this sequence:

   ```
   Security & Users
      Groups
         Add a Group
   ```

   You can take the default values for the attributes of the new group or change them as required.

2. Create the new user, `mqm`. You can display the window for doing this using this sequence:

   ```
   Security & Users
      Users
         Add a User
   ```

Set the primary group for this user to be mqm. You can take the default values for the attributes of the new group or change them if you wish.

3. Add a password to the new user ID. You can display the window for doing this using this sequence:

   ```
   Security & Users
      Passwords
   ```

4. Add the newly created group mqm to an existing user ID. You can display the window for doing this using this sequence:

   ```
   Security & Users
      Users
         Change / Show Characteristics of a User
   ```

   When the window is displayed, enter the name of the user who is to have the mqm group added. In the user details window, add mqm to the **Group set** field, which is a comma-separated list of the groups to which the user belongs.

Note: Users need not have their primary group set to mqm. As long as mqm is in their set of groups, they can use the commands. Users who are running applications that use the queue manager only do not need mqm group authority.

Easy installation

1. Logon as root.
2. Insert the MQSeries Client CD-ROM into the CD-ROM drive.
3. Type `xinstallm -ez`

   The MQSeries Welcome window is displayed.
4. ***Make sure you are installing the correct client*** for your system, as displayed in the Welcome window.

   A window is then displayed where you can make some choices.
5. Choose the software source: **CD-ROM**.
6. For **Which bundle of software would you like to install?** choose: **Media-defined**.
7. Click on **Install/Update**.

   A bundle of software products is created:

   ```
   mqm.Client
   ```

   See “Components for UNIX platforms” on page 32 for details of the components.
8. Choose the mqm.Client bundle and click on **Install/Update** again.

   A work in progress window gives information as the installation proceeds.
9. At the end of installation you can click on the **View log** button and scroll to the bottom of the log to see the filesets that have been installed successfully.

Now go to Chapter 5, “Verifying the installation” on page 65.

Custom installation

You can use SMIT for a custom installation as follows:

1. Logon as root.
2. Go into SMIT and from the shell, type:

   ```
   smitty
   ```

3. Select the device appropriate for your installation using this sequence of windows:

   ```
     Software Installation & Maintenance
        Install and Update Software
           Install/Update Selectable Software (Custom Install)
              Install Software Products at Latest Level
                 Install New Software Products at Latest Level
   ```

   You can use the alternative fastpath command instead:

   ```
   smitty install_latest
   ```

   Press the **List** button to display the **Single Select List** window.

   Select:

   `/dev/cd0 (CD-ROM Drive)`

4. Press **Do** to display the parameters for **Install Latest Level**.
5. Press **F4** to get a list of components to install.
6. Follow the SMIT instructions to select the components you want to install. See "Components for UNIX platforms" on page 32 for details.
7. Press **Enter**.
8. If you have a previous version of the product on your machine, change the **Auto Install prerequisite software** to **No** and **Overwrite existing version** to **Yes**.
9. Press **Do** to install.

Now go to Chapter 5, "Verifying the installation" on page 65.

Changing the national language

The easy installation and the custom installation default to the national language that was specified when your operating system was installed.

It is possible to install the MQSeries client software so that the online help and messages are in another national language. Use SMIT as follows to change to another national language and then install (or reinstall) the MQSeries client using SMIT as detailed above.

1. Select `Install/Update From All Available Software`.
2. Press **F4** to see a choice of national languages.
3. You are ***strongly*** recommended to select the en_US national language as well as any others that you want to install. If you do not do this and your locale setting is not found in one of the message catalogs, no message catalogs will be installed.
4. Press **Do** to install the chosen message catalog or catalogs.

To check the initial locale setting for your machine type:

```
smitty mle_cc_cust_hdr
```

and press the space bar. If this is not one of the national languages provided by MQSeries, you need to select a national language, otherwise you will not get a message catalog installed on your system.

Removing an MQSeries client from AIX

Use SMIT as usual to remove all the MQSeries client files that were installed.

Installing on DOS

To install an MQSeries client on a DOS system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Using Setup

1. Insert the MQSeries Client CD-ROM into the CD-ROM drive.
2. Change to the DOS directory on the CD-ROM drive.
3. Copy the **setup.exe** file from the DOS directory to the directory where you want to install the MQSeries client, for example C:\mqmdos.
4. Change directory to C:\mqmdos and type the command:

   ```
   setup -d
   ```

   This results in a self-exploding file being run to generate a tree of sub-directories containing the DOS client. See “Components for DOS and Windows 3.1” on page 34 for details.
5. Edit the autoexec.bat file using a suitable editor. If the PATH statement exists, add the following to it:

   ```
   c:\mqmdos;c:\mqmdos\bin;c:\mqmdos\en_us
   ```

   If the PATH statement does not exist, add the following line to the autoexec.bat file.

   ```
   SET PATH=c:\mqmdos;c:\mqmdos\bin;c:\mqmdos\en_us
   ```

Now go to Chapter 5, “Verifying the installation” on page 65.

Removing an MQSeries client from DOS

Delete all the files in the directory where you installed the MQSeries client, and then remove the directory.

Installing on HP-UX

To install an MQSeries client on an HP-UX system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Note: If you plan to install an MQSeries client and server on the same machine, see “Installing MQSeries clients on the same machine as the server” on page 35.

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

Before installation

Before you can install an MQSeries client on your HP-UX system you:

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

  If you create separate partitions, the following directories ***must*** be on a local file system:

  – `/var/mqm`
  – `/var/mqm/log`
  – `/var/mqm/objects`

  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

Use the HP-UX **swinstall** program, or use SAM, after mounting the CD-ROM. For further details see the *HP-UX Administration Guide*. See also “Components for UNIX platforms” on page 32.

| The component **MQSeries Client for Java** should be installed only if you have
| Java 1.1.2 (or later compatible) runtime code on your machine. Also you must use
| HP-UX Version 10.20.

| The **depot** to use is then in the `HPUX10/MQS500.000` file under the mount point.

If the files on your CD-ROM appear in uppercase with a “;1” suffix, use this name for the depot.

Kernel configuration

See the MQSeries family homepage for a SupportPac that gives additional performance information - see “MQSeries information available on the Internet” on page xiii.

Now go to Chapter 5, “Verifying the installation” on page 65.

Translated messages

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

For example, to select messages in German use the following:

```
export LANG=de_De.iso88591
export NLSPATH=/usr/lib/nls/msg/%L/%N
```

Removing an MQSeries client from HP-UX

To remove an MQSeries client from your HP-UX system, use the **swremove** command, or use SAM. You can then delete the /var/mqm directory tree.

Installing on OS/2 Warp

To install an MQSeries client on an OS/2 Warp system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Note: If you plan to install an MQSeries client and server on the same machine, see “Installing MQSeries clients on the same machine as the server” on page 35.
| If you currently have a manually installed MQSeries client on your OS/2 Warp
| system from a previous release of MQSeries, you must manually delete it before
| attempting to install the Version 5 client. You must not install the Version 5 client
| onto a system which currently has a Version 2 MQSeries Server installed.

You can install the version of the MQSeries client software specific to your national language. This means that the installation program, online help and messages will be in your national language.

Installation

Online help is available by selecting the **Help** push button or by pressing PF1.

Before you start, make sure that you have at least 150 KB of free space on the drive containing the operating system. This is required by the installation program.

1. Open an OS/2 window (or start a full-screen session).
2. Insert the CD-ROM and change to the CD-ROM drive. Access the drive and directory containing the installation program if you are installing from a remote drive:
3. At the command prompt, in the root directory, type `INSTALL`, then press Enter.
4. On the MQSeries Language Selection panel select the language of your choice, and click on the **OK** button or press Enter.

   The MQSeries Welcome panel is displayed. ***Make sure you are installing the correct client*** for your system, OS/2 Warp, as displayed in the Welcome panel.
5. The install panel is then displayed. Select the **Update CONFIG.SYS** check box if you want your CONFIG.SYS file updated automatically as part of the installation process. Your original CONFIG.SYS file is renamed to CONFIG.BAK and is stored in the same directory. If you do not select this check box, a CONFIG.ADD file is generated. This file is a copy of CONFIG.SYS with the necessary updates to the LIBPATH and PATH statement. You can rename the CONFIG.ADD file to CONFIG.SYS.

6. Select the **OK** push button to continue. The Install - directories panel is displayed.
7. The list box shows the installation options that you can select. When you select one or more of these options, the **Bytes needed** field shows the amount of disk space required for installation. See “Components for OS/2 Warp and Windows NT” on page 33 for details.

   | The component **MQSeries Client for Java** should be installed only if you have
   | Java 1.1.1 (or later compatible) runtime code on your machine. Also this
   | component must be installed on an HPFS formatted drive
8. If there is not enough space on your hard disk to install all the components, select an option that uses less disk space. If there is too little space on your hard disk for any of the MQSeries for OS/2 installation options, a dialog box appears before the Install - directories panel. In this case, cancel the installation by selecting the **OK** push button. Find out which of your existing files you can archive or delete to make more space before proceeding further.

   Use the push buttons as necessary:

   - To display descriptions of the selected options, select **Descriptions**.
   - To select all of the options, select **Select all**.
   - To deselect all of the options, select **Deselect all**.

   The **Work and File Directory** field allows you to specify a drive and directory other than the default for the installation files (File directory) and for the working files that may be created when you use the MQSeries client (Working directory).

   Select a drive from the list box if required. When you return to the Install - directories panel, your selected drive is shown. Select the **OK** push button to return to the Install - directories panel.
9. Select the **Install** push button to continue. The Install-progress panel is displayed. The panel shows:

   - The file currently being installed (source) and the drive and directory to which it is being installed (target).
   - A progress bar, indicating the percentage of files already unpacked and installed.
   - The elapsed time.
   - The status; for example, unpacking, processing, or transferring.

   If you select the **Stop** push button, you are asked whether you want to delete the partial system you have installed. Select **Yes** to delete the files already installed and return to the introductory panel. Then, select **Start install** from the **File** menu to start the installation again.
10. A cyclic redundancy check (CRC) is performed on the installed software and any errors are written to a log file. This is the file specified by the /L1 parameter of the INSTALL command by default. If /L1 is not specified, the log file is MQMERR.LOG in the high-level directory chosen for installation.

    Note: The log files ***must*** be on a local drive. If the product has been installed on a remote drive, change the path of the log files in the mqs.ini file.

11. When installation is complete, the Installation and Maintenance panel is displayed. Select **OK**. The introductory MQSeries for OS/2 panel is then displayed. Leave the installation program by selecting the **Exit** push button.
12. When the installation process is complete, a folder is created on the OS/2 desktop, containing objects as follows:

    READ.ME
    MQSeries Installation and Maintenance
    MQSeries Information

    Note that the MQSeries client is a set of services and it does not have to be explicitly run. Therefore the folder does not have an object called a “client”.
13. Remove the installation CD-ROM from the drive.
14. If your CONFIG.SYS file has been updated, shut down the system and restart. If the CONFIG.SYS file was not updated, rename the CONFIG.ADD file to CONFIG.SYS before shutting down the system. (CONFIG.ADD will be in the same directory as CONFIG.SYS.)

Now go to Chapter 5, “Verifying the installation” on page 65.

Unattended installation

It is possible to install MQSeries clients using an unattended installation method. For details of this see the book *MQSeries for OS/2 Warp V5.0 Quick Beginnings*.

Removing an MQSeries client from OS/2

Use the MQSeries Installation and Maintenance icon in the MQSeries Client folder on the desktop, and select Actions/Delete. All the MQSeries client files that were there at the time of installation are deleted.

Installing on Sun Solaris

To install an MQSeries client on a Sun Solaris system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Note: If you plan to install an MQSeries client and server on the same machine, see “Installing MQSeries clients on the same machine as the server” on page 35.

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 Sun Solaris system you:

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

  If you create separate partitions, the following directories ***must*** be on a local file system:

  – `/var/mqm`
  – `/var/mqm/log`

   – `/var/mqm/objects`

  You can choose to NFS mount the `/var/mqm/errors` and `/var/mqm/trace` directories to conserve space on your local system.

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

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

Installation

Carry out the following procedure:

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

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

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

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

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

2. Use the Sun Solaris **pkgadd** program, to install the MQSeries client software by carrying out the following procedure:

   a. Type `pkgadd -d /cdrom/mqclient/solaris/mqs500.img`.

   b. You are prompted for a list of components to be installed. Select the ones you require - if you want to install all the components, select **all**.

      See “Components for UNIX platforms” on page 32 for details.

      The component **MQSeries Client** for Java should be installed only if you have Java 1.1.1 (or later compatible) runtime code on your machine. Also you require version 2.5.1 of the Solaris Operating System plus the following patches: 103566-08, 103600-13, 103640-08.

   c. Press the Enter key.

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

Kernel configuration

See the MQSeries family homepage for a SupportPac that gives additional performance information - see “MQSeries information available on the Internet” on page xiii.

Now go to Chapter 5, “Verifying the installation” on page 65.

Translated messages

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

For example:

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

Removing an MQSeries client from Sun Solaris

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

If the product is present, but not installed correctly, you may need manually to delete the files and directories contained in:

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

Installing on Windows 3.1

To install an MQSeries client on a a Windows 3.1 system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Installation

1. Insert the MQSeries Client CD-ROM into the CD-ROM drive.
2. Open the directory `WIN31`.
3. Change to the appropriate language subdirectory for the language you wish to install:

   ```
   setupen - English
   setupfr - French
   setupde - German
   setupes - Spanish
   setupit - Italian
   setuppt - Portuguese Brazilian
   setupjp - Japanese
   setupko - Korean
   setupcn - Simplified Chinese
   setuptw - Traditional Chinese
   ```
4. Run setup.exe

   The MQSeries Welcome window is displayed.
5. ***Make sure you are installing the correct client*** for your system, Windows 3.1, as displayed in the Welcome window.
6. Select Destination Directory requires a destination directory into which the MQSeries files will be installed.

   You can change the default directory by selecting the browse button and choosing a different drive and directory, then click on **OK**. Click on the **Next** button or press Enter to continue.

7. Choose MQSeries Components displays a list of components from which you can select the ones you want to be installed. See “Components for DOS and Windows 3.1” on page 34 for details.

   To select a component, click in the square next to it so that a check mark appears (just highlighting the line does not select it). The space needed for each component is shown here, and the space available on the drive you have selected. Click on the **Next** button or press Enter to continue.

8. Select Program Folder prompts you for a folder name to contain the MQSeries objects. The default name is MQSeries Client for Windows 3.1. You can rename the default or choose an existing folder.

9. Start Copying Files displays the selection you have made. Click on the **Back** button if you want to return to change your choice.

   Now click on the **Next** button or press Enter to start the file copying process.

   The progress indicator shows what components are being copied and the percentage of copying completed.

10. The next window presents you with the opportunity to view the README file. If you do not want to view the README file at this point, it will be available in the MQSeries client folder.

    If you view the README file, close the window of the README to continue the installation process.

11. The installation of the MQSeries client is now complete, and a window is displayed giving you the option of restarting your computer now or leaving it until later. It is recommended that you restart your computer now. Close all the other applications that are running before continuing with this step.

    Select `Yes, I want to restart my computer now` and click on the **Finish** button to complete the setup.

12. When setup is complete, the MQSeries Client folder is added to the Program Manager. Note that the MQSeries client is a set of services and it does not have to be explicitly run, so the folder does not have an object called a “client”.

Running Setup again

You can run the installation again to add another component or to reinstall a component. If you want to reinstall a component you must first remove it, see “Removing an MQSeries client from Windows 3.1.”

If components are already installed and you cancel the reinstallation before any files have been copied, you will see the message `Setup is not complete`. This means that nothing has been done, so the installation remains as before.

Removing an MQSeries client from Windows 3.1

If you want to remove the MQSeries client files from your machine, use the process provided to do this efficiently.

Run `Uninstall` from the MQSeries client folder. You are prompted before continuing.

All the MQSeries client files that were created at installation time are removed by the process.

Installing on Windows NT or Windows 95

To install an MQSeries client on a Windows NT or a Windows 95 system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Note: If you plan to install an MQSeries client and server on the same machine, see "Installing MQSeries clients on the same machine as the server" on page 35.

| **Note for Windows NT users:** If you have the IBM Anti-Virus software installed on
| your Windows NT workstation/Server Version 4 and you install IBM MQSeries, the
| folders containing the MQSeries programs and sub-folders will not be created. The
| solution to this problem is to de-install IBM Anti-Virus software, install MQSeries
| and then re-instate IBM Anti-Virus software.

Installation

1. Insert the MQSeries Client CD-ROM into the CD-ROM drive.
2. The installation automatically launches and an MQSeries Language Selection window is displayed.

   Note: If you have disabled auto-playing of CD-ROMs, run SETUP instead, from the root directory.

   This window presents you with a list of the National Languages that are available.
3. On the MQSeries Language Selection window select the language of your choice, and click on the **Next** button or press Enter.

   The MQSeries Welcome window is displayed.
4. ***Make sure you are installing the correct client*** for your system, as displayed in the Welcome window.
5. Select Destination Directory requires a destination directory into which the MQSeries files will be installed.

   You can change the default shown by selecting the browse button and choosing a different drive and directory, then click on **OK**. Click on the **Next** button or press Enter to continue.
6. Choose MQSeries Components displays a list of components from which you can select the ones you want to be installed. See "Components for OS/2 Warp and Windows NT" on page 33 or "Components for Windows 95" on page 34 for details.

   | The component **MQSeries Client for Java** should be installed only if you have
   | Java 1.1.1 (or later compatible) runtime code on your machine. Also on a
   | Windows NT client you must use Microsoft Windows NT Version 4.

   To select a component, click in the square next to it so that a check mark appears (just highlighting the line does not select it). The space needed for each component is shown here, and the space available on the drive you have selected.

   Click on the **Next** button or press Enter to continue.
7. Select Program Folder prompts you for a folder name to contain the MQSeries objects. The default name is MQSeries Client for Windows NT (or 95). You can rename the default or choose an existing folder.

8. Start Copying Files displays all the selections you have made. Click on the **Back** button if you want to return to a previous window to change any of your choices.

   When you have checked your choices, Click on the **Next** button or press Enter to start the file copying process.

   The progress indicator shows which components are being copied and the percentage of copying completed.

9. The next window presents you with the opportunity to view the README file. If you do not want to view the README file at this point, it will be available in the MQSeries client folder.

   If you view the README file, close the window of the README to continue the installation process.

10. The installation of the MQSeries client is now complete, and a window is displayed giving you the option of restarting your computer now or leaving it until later. It is recommended that you restart your computer now. Close all the other applications that are running before continuing with this step.

    Once this has been done, select `Yes, I want to restart my computer now` and click on the **Finish** button to complete the setup.

11. When setup is complete, the MQSeries Client folder is added to the desktop, or the location you specified. Note that the MQSeries client is a set of services and it does not have to be explicitly run, so the folder does not have an object called a “client”.

Silent install on Windows 95 and Windows NT

1. Change to the required language subdirectory, for example `f:\winnt\setupen` (where f is your CD-ROM drive).

2. Run setup.exe with the -r option:

   ```
   setup.exe -r
   ```

   This does a recorded install of MQSeries to your machine. The process records this to file setup.iss in the subdirectory c:\WINNT (on Windows NT) or in the subdirectory c:\WINDOWS (on Windows 95).

3. This .iss file needs to be placed in the language subdirectory. For example copy all files from f:\winnt\ to s:\winnt\ and copy setup.iss to s:\winnt\setupen\setup.iss (where s is the shared resource).

4. On the machine you wish to silently install, connect to this shared resource. Then from s:\winnt\setupen run setup.exe with the -s option, for example:

   ```
   setup.exe -s
   ```

   This should now silently install the client to this machine.

Now go to Chapter 5, “Verifying the installation” on page 65.

Running Setup again

You can run the installation again to add another component or to reinstall a component. If you want to reinstall a component you must first remove it, as described in “Removing an MQSeries client from Windows NT and Windows 95.”

If components are already installed and you cancel the reinstallation before any files have been copied, you will see the message `Setup is not complete`. This means that nothing has been done, so the installation remains as before.

Removing an MQSeries client from Windows NT and Windows 95

If you want to remove the MQSeries client files from your machine, use Settings/ Control Panel/ Add-Remove programs. First select MQSeries Client which launches the uninstall program. Then select the components you want to remove and click **Remove**.

If you choose to remove all MQSeries components and are then likely to reinstall MQSeries, you should restart your computer to complete the uninstall process. You cannot reinstall any components until you have restarted.

All the MQSeries client files that were created at installation time are removed by the process.

Chapter 4. Installing MQSeries clients with non-Version 5 products

The MQSeries non-Version 5 products are:

- MQSeries for AS/400 V4R2
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Digital OpenVMS V2.2
- MQSeries for MVS/ESA V1.2
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for SunOS V2.2
- MQSeries for Tandem NonStop Kernel V2.2
- MQSeries Client for VM/ESA V2.3

If you are using an MQSeries Version 5 product, see Chapter 3, “Installing MQSeries client components from Version 5 products” on page 31.

You install the MQSeries client and server system in two parts: one for the MQSeries server on your server machine and one for the MQSeries client on your client machine.

You can install the MQSeries client files from your MQSeries product, as explained here, or from an IBM Transaction Processing SupportPac, see “MQSeries clients from IBM Transaction Processing SupportPacs.”

Note for AS/400, MVS/ESA, and Tandem NSK users: MQSeries for AS/400 V4R2, MQSeries for MVS/ESA V1.2, and MQSeries for Tandem NonStop Kernel V2.2 can accept connections from MQSeries clients on other platforms. An MQSeries client cannot run on AS/400, MVS/ESA, or Tandem NSK and the files for MQSeries clients are not supplied with these products.

If you want to connect MQSeries clients with these platforms, install the MQSeries clients from another MQSeries product, or from an IBM Transaction Processing SupportPac (see “MQSeries clients from IBM Transaction Processing SupportPacs”).

Note for VM/ESA users: You do not need the installation procedure detailed for the other platforms. See “MQSeries Client for VM/ESA Version 2 Release 3” on page 63.

MQSeries clients from IBM Transaction Processing SupportPacs

The MQSeries client files can be copied from the IBM Transaction Processing SupportPacs for use as needed. See “MQSeries information available on the Internet” on page xiii.

The IBM Transaction Processing SupportPacs library consists of material that complements the family of CICS and MQSeries products marketed by IBM.

MQSeries client software is available at no charge and is subject to the IPLA and License Information terms defined when requesting the MQSeries clients on the Internet. You have the right to make as many copies of the MQSeries client as necessary.

MQSeries clients from the MQSeries products

The way you install the MQSeries client files depends on whether your client platform is the same as, or different from, the server platform.

MQSeries client and server on the same platform

If your MQSeries client platform is the same as your server platform, you can install both of them in the normal way, directly from the media (diskette, CD-ROM, or tape, according to your platform). For details see the *System Management Guide* for your platform.

You can, instead, install the MQSeries client from a LAN server on which you have already installed the Base product and server (not recommended for an MQSeries client on Windows NT). Use FTP or a similar method to do this. For NetView instructions, see the *System Management Guide* for your platform.

Full details of how to copy the required MQSeries client files to the client machine, including the filenames and directories, are given in this chapter:

Installing MQSeries clients from:

- MQSeries for Digital OpenVMS (page 54)
- MQSeries for UNIX systems (page 58)

MQSeries client and server on different platforms

If your MQSeries client platform is different from your server platform, you may only require the MQSeries product for the server platform. The MQSeries Version 2.2 products supply files for a group of *desktop clients*. These are in addition to the files for the MQSeries client on the same platform as your server. The desktop clients are for DOS, OS/2 and Windows 3.1.

You install MQSeries on the server machine in the normal way directly from the media (diskette, CD-ROM or tape, according to your platform). You may also install, at the same time, the MQSeries client files that you need for your other platform or platforms.

Then you copy the required MQSeries client files to the build and run environment on the client machine. Full details of how to do this, including the file names and directories, are given in this chapter:

Installing MQSeries clients from:

- MQSeries for Digital OpenVMS (page 54)
- MQSeries for UNIX systems (page 58)

Installing the MQSeries server

Install the MQSeries Base Product and Server on the machine you want to use as your MQSeries server. Full details are given in the *System Management Guide* for your platform.

For MQSeries for MVS/ESA, install the MQSeries Base product and Distributed Queuing without CICS feature, and Client Attachment feature, on the machine you want to use as your MQSeries server. Customize it as required to provide client-server support. Full details are given in the *MQSeries for MVS/ESA Program Directory*.

MQSeries client files on the server

During the installation of the server you may be able to include MQSeries client files for other platforms. This depends on your server platform, as follows.

MQSeries for Digital OpenVMS, and UNIX platforms

When you install MQSeries on Digital OpenVMS, or a UNIX system, the files for some desktop clients are supplied: DOS, OS/2, and Windows 3.1. Also files for MQSeries clients on the same platform as the server are supplied.

On the server, on the appropriate menu during the installation, you select the MQSeries client or clients that you require. Then you continue with the installation of the server. The MQSeries client files will be copied onto the server, ready for you to copy onto your client machine, as described in this chapter.

MQSeries for AS/400, MVS/ESA, and Tandem NSK

| You can connect MQSeries clients on other platforms to MQSeries for AS/400
| V4R2, MQSeries for MVS/ESA V1.2, and MQSeries for Tandem NonStop Kernel
| V2.2. MQSeries clients cannot be run on these systems, so the files for MQSeries
| clients are not supplied with MQSeries for AS/400 V4R2, MQSeries for MVS/ESA
| V1.2, or MQSeries for Tandem NonStop Kernel V2.2.

If you have only these MQSeries products, and you want to install MQSeries clients on other platforms, see “MQSeries clients from IBM Transaction Processing SupportPacs” on page 51.

If you have another MQSeries product available, you may be able to install the MQSeries client files you require from that product, as described in the following sections of this chapter.

Installing MQSeries clients from MQSeries for Digital OpenVMS

When you have included the MQSeries client files in the installation of your Digital OpenVMS server machine, the files are located in these directories:

Digital OpenVMS files

```
SYS$LIBRARY
SYS$SYSTEM
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES]
```

OS/2 files

```
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.BIN]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.DLL]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.INC]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.LIB]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.MSG]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.SAMP.BIN]
```

DOS files

```
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.DOS_CLIENT.BIN]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.DOS_CLIENT.INC]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.DOS_CLIENT.LIB]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.DOS_CLIENT.MSG]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.DOS_CLIENT.SAMP]
```

Windows 3.1 files

```
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.BIN]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.DLL]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.INC]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.LIB]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.MSG]
SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.SAMP.BIN]
```

Note: This assumes that the logical name `MQS_EXAMPLES` is assigned to `SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES]`

Installing an MQSeries client on Digital OpenVMS from Digital OpenVMS

If possible do this by installing the MQSeries client directly from the media supplied with your MQSeries product. For details see *MQSeries for Digital OpenVMS System Management Guide*.

You can, instead, use the following method:

1. Copy these files from the the Digital OpenVMS server to the Digital OpenVMS client system, into the same directories as on the server system:

```
SYS$COMMON:[SYSEXE]DSPMQTRC.EXE
SYS$COMMON:[SYSEXE]ENDMQTRC.EXE
SYS$COMMON:[SYSEXE]RUNMQTRC.EXE
SYS$COMMON:[SYSEXE]STRMQTRC.EXE

SYS$COMMON:[SYSLIB]AMQCC62A.EXE
SYS$COMMON:[SYSLIB]AMQCCDCA.EXE
SYS$COMMON:[SYSLIB]AMQCCTCA.EXE
SYS$COMMON:[SYSLIB]AMQTRC.FMT
SYS$COMMON:[SYSLIB]MQIC.EXE
SYS$COMMON:[SYSLIB]MQICB.EXE
SYS$COMMON:[SYSLIB]MQMCS.EXE
```

2. Creat this directory on the Digital OpenVMS client system:

   ```
   SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES]
   ```

   Copy into this directory all the files from the same directory on the Digital OpenVMS server.

Installing an MQSeries client on OS/2 from Digital OpenVMS

1. Create this directory structure on the OS/2 system:

   ```
   <drive>:\mqm\bin
   <drive>:\mqm\dll
   <drive>:\mqm\inc
   <drive>:\mqm\lib
   <drive>:\mqm\msg
   ```

2. Copy these files, from Digital OpenVMS, into the above directories on the OS/2 system:

   ```
   <drive>:\mqm\bin

   SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.BIN]
   SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.SAMP.BIN]

   <drive>:\mqm\dll

   SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.DLL]

   <drive>:\mqm\inc

   SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.INC]
   <drive>:\mqm\lib

   SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.LIB]

   <drive>:\mqm\msg

   SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.OS2_CLIENT.MSG]
   ```

   See Chapter 5, “Verifying the installation” on page 65 for how to use the sample executables amqsputc.exe and amqsgetc.exe. Alternatively, you can build these yourself from the corresponding source files amqsput0.c and amqsget0.c.

3. Create an MQSeries configuration file (mqs.ini) in the `<drive>:\mqm` directory and add the following stanza:

```
AllQueueManagers:
  DefaultPrefix=<drive>:\mqm
```

4. Make the following changes to your config.sys file on the OS/2 system (for details of how to make the changes, see "Changing the OS/2 config.sys file" on page 62):

   ```
   PATH:     include <drive>:\mqm\bin
   DPATH:    include <drive>:\mqm\bin;c:\mqm\msg
   LIBPATH:  include <drive>:\mqm\dll
   HELP:     include <drive>:\mqm\bin
   LIB:      include <drive>:\mqm\lib
   ```

   Reboot your system for these changes to take effect.

5. Now go to Chapter 5, "Verifying the installation" on page 65.

Installing an MQSeries client on DOS from Digital OpenVMS

1. Create a suitable directory on the DOS system.

2. Copy these files, from Digital OpenVMS `SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.DOS_CLIENT]` into the directory you have created on the DOS system:

   ```
     *.exe
     *.msg
     *.lib
     *.h
   ```

   See Chapter 5, "Verifying the installation" on page 65 for how to use the sample executables (included when you copy *.exe): amqsputc.exe and amqsgetc.exe. Alternatively, you can build these yourself from the corresponding source files amqsput0.c and amqsget0.c, in `<drive>:\mqm\tools\c\samples`.

3. Make changes to the PATH statement and the DOS APPEND statement in your autoexec.bat file to include the directory you have used (for details of how to make the changes, see "Changing the autoexec.bat file for DOS and Windows 3.1" on page 63).

4. Now go to Chapter 5, "Verifying the installation" on page 65.

Installing an MQSeries client on Windows 3.1 from Digital OpenVMS

Note: You can also install the Windows 3.1 client code on WIN-OS/2 under OS/2, Windows NT, and Windows 95.

1. Create a suitable directory on the Windows 3.1 system.

2. Copy these files, from MQSeries for Digital OpenVMS `SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.BIN]` into the directory you have created on the Windows 3.1 system:

   ```
     *.exe
     *.dll
     *.msg
     *.lib
     *.h files
   ```

   See Chapter 5, "Verifying the installation" on page 65 for how to use the sample executables (included when you copy *.exe): `amqsputw.exe` and `amqsgetw.exe`. Alternatively, you can build these yourself from the corresponding source files `amqsputw.c` and `amqsgetw.c`, in `<drive>:\mqm\win`.

3. Make changes to the PATH statement and the DOS APPEND statement in your autoexec.bat file to include the directory you have used (for details of how to make the changes, see “Changing the autoexec.bat file for DOS and Windows 3.1” on page 63).
4. Now go to Chapter 5, “Verifying the installation” on page 65.

Installing MQSeries clients from MQSeries for UNIX systems

Installation directory for UNIX systems:
The name **mqmtop** is used to represent the name of the installation directory for UNIX systems.

The name of the actual directory is:

MQSeries for SunOS	/usr/mqm
MQSeries for AT&T GIS UNIX MQSeries for SINIX and DC/OSx	/opt/mqm

When you have included the MQSeries client files in the installation of your UNIX system server machine, the files are located in these directories:

UNIX system files

```
/mqmtop/xxx_client/bin
/mqmtop/xxx_client/inc
/mqmtop/xxx_client/lib
/mqmtop/xxx_client/msg
/mqmtop/xxx_client/samp/bin
```

Where `xxx` is an identifier for the name of the UNIX system on your server machine.

OS/2 files

```
/mqmtop/os2_client/bin
/mqmtop/os2_client/dll
/mqmtop/os2_client/inc
/mqmtop/os2_client/lib
/mqmtop/os2_client/msg
/mqmtop/os2_client/samp/bin
```

DOS files

```
/mqmtop/dos_client/bin
/mqmtop/dos_client/lib
/mqmtop/dos_client/inc
/mqmtop/dos_client/msg
/mqmtop/dos_client/samp/bin
```

Windows 3.1 files

```
/mqmtop/win_client/bin
/mqmtop/win_client/dll
/mqmtop/win_client/inc
/mqmtop/win_client/lib
/mqmtop/win_client/msg
/mqmtop/win_client/samp/bin
```

Installing an MQSeries client on a UNIX system from a UNIX system

If possible do this by installing the MQSeries client directly from the media supplied with your MQSeries product. For details see the *System Management Guide* for your platform.

You can, instead, use the following method:

1. Create this directory structure on the UNIX system:

   ```
   /mqmtop/mqm/bin
   /mqmtop/mqm/inc
   /mqmtop/mqm/lib
   /mqmtop/mqm/msg
   ```

2. Copy these files, from the server UNIX system, into the above directories on the client UNIX system:

   ```
   /mqmtop/mqm/bin
     /mqmtop/xxx_client/bin
     /mqmtop/xxx_client/samp/bin/amqsputc.exe
     /mqmtop/xxx_client/samp/bin/amqsgetc.exe

   /mqmtop/mqm/inc
     /mqmtop/xxx_client/inc

   /mqmtop/mqm/lib
     /mqmtop/xxx_client/lib

   /mqmtop/mqm/msg
     /mqmtop/xxx_client/msg
   ```

   See Chapter 5, “Verifying the installation” on page 65 for how to use the sample executables amqsputc.exe and amqsgetc.exe. Alternatively, you can build these yourself from the corresponding source files amqsput0.c and amqsget0.c.

   Note: For national language-specific directories, see the `README` file supplied with MQSeries on the server platform.

3. Copy the file amq.cat from **/mqmtop**`/mqm/msg` into the default message catalog directory for your system, or a directory referenced in your NLSPATH environment variable.

   For example:

On AT&T GIS UNIX	`/usr/lib/locale/C/LC_MESSAGES`

4. Make a directory `/var/mqm`. Create an MQSeries configuration file (mqs.ini) in the `/var/mqm` directory and add the following stanza:

   ```
   AllQueueManagers:
     DefaultPrefix=/var/mqm
   ```

5. This step is optional:

   Symbolically link the C header files from **mqmtop**`/mqm/inc` into `/usr/include`

   Link the libraries from **mqmtop**`/mqm/lib` into `/usr/lib`

   Link the programs from **mqmtop**`/mqm/bin` into `/usr/bin`

6. Now go to Chapter 5, “Verifying the installation” on page 65.

Installing an MQSeries client on OS/2 from a UNIX system

1. Create this directory structure on the OS/2 system:

```
<drive>:\mqm\bin
<drive>:\mqm\dll
<drive>:\mqm\inc
<drive>:\mqm\lib
<drive>:\mqm\msg
<drive>:\mqm\samp\bin
```

2. Copy these files, from MQSeries on the UNIX system, into the above directories on the OS/2 system:

   ```
   <drive>:\mqm\bin
     /mqmtop/os2_client/bin

   <drive>:\mqm\dll
     /mqmtop/os2_client/dll

   <drive>:\mqm\inc
     /mqmtop/os2_client/inc

   <drive>:\mqm\lib
     /mqmtop/os2_client/lib

   <drive>:\mqm\msg
     /mqmtop/os2_client/msg

   <drive>:\mqm\samp\bin
     /mqmtop/os2_client/samp/bin/amqsputc.exe
     /mqmtop/os2_client/samp/bin/amqsgetc.exe
   ```

   See Chapter 5, “Verifying the installation” on page 65 for how to use the sample executables amqsputc.exe and amqsgetc.exe. Alternatively, you can build these yourself from the corresponding source files amqsput0.c and amqsget0.c.

   Note: For national language-specific directories, see the README file supplied with MQSeries on the server platform.

3. Create an MQSeries configuration file (mqs.ini) in the `<drive>:\mqm` directory and add the following stanza:

   ```
   AllQueueManagers:
     DefaultPrefix=<drive>:\mqm
   ```

4. Make the following changes to your config.sys file on the OS/2 system (for details of how to make the changes, see “Changing the OS/2 config.sys file” on page 62):

   ```
   PATH:     include <drive>:\mqm\bin
   DPATH:    include <drive>:\mqm\bin;c:\mqm\msg
   LIBPATH:  include <drive>:\mqm\dll
   HELP:     include <drive>:\mqm\bin
   LIB:      include <drive>:\mqm\lib
   INCLUDE:  include <drive>:\mqm\inc
   ```

5. Now go to Chapter 5, “Verifying the installation” on page 65.

Installing an MQSeries client on DOS from a UNIX system

1. Copy these files, from MQSeries on the UNIX system into a suitable directory on the DOS system:

   ```
   /mqmtop/dos_client/bin
     *.exe

   /mqmtop/dos_client/inc
     *.h

   /mqmtop/dos_client/lib
     *.lib

   /mqmtop/dos_client/msg
     *.msg

   /mqmtop/dos_client/samp/bin/amqsputc.exe
   /mqmtop/dos_client/samp/bin/amqsgetc.exe
   ```

   See Chapter 5, “Verifying the installation” on page 65 for how to use the sample executables amqsputc.exe and amqsgetc.exe. Alternatively, you can build these yourself from the corresponding source files.

   Note: For national language-specific directories, see the `README` file supplied with MQSeries on the server platform.

2. Make changes to the PATH statement and the DOS APPEND statement in your autoexec.bat file to include the directory you have used (for details of how to make the changes, see “Changing the autoexec.bat file for DOS and Windows 3.1” on page 63).

3. Now go to Chapter 5, “Verifying the installation” on page 65.

Installing an MQSeries client on Windows 3.1 from a UNIX system

Note: You can also install the Windows 3.1 client code on WIN-OS/2 under OS/2, Windows NT, and Windows 95.

1. Copy these files, from MQSeries on the UNIX system into a suitable directory, for example `c:\mqm` on the Windows system:

   ```
   /mqmtop/win_client/bin
     *.exe

   /mqmtop/win_client/dll
     *.dll

   /mqmtop/win_client/lib
     *.lib

   /mqmtop/win_client/msg
     *.msg

   /mqmtop/win_client/inc/
     *.h
     *.hpp  (Version 5 products)

   /mqmtop/win_client/samp/bin/amqsputw.exe
   /mqmtop/win_client/samp/bin/amqsgetw.exe
   /mqmtop/win_client/samp/bin/imq*c.exe  (Version 5 products)
   ```

See Chapter 5, “Verifying the installation” on page 65 for how to use the sample executables `amqsputw.exe` and `amqsgetw.exe`. Alternatively, you can build these yourself from the corresponding source files.

Note: For national language-specific directories, see the `README` file supplied with MQSeries on the server platform.

2. Make changes to the PATH statement and the DOS APPEND statement in your autoexec.bat file to include the directory you have used (for details of how to make the changes, see “Changing the autoexec.bat file for DOS and Windows 3.1” on page 63).
3. Now go to Chapter 5, “Verifying the installation” on page 65.

Changing config.sys and autoexec.bat

On OS/2, Windows 3.1, and DOS systems, you need to add some statements to the files that the operating system uses when it starts up. The files are config.sys and autoexec.bat. These statements define to the operating system the path or paths to the MQSeries client directories.

The way you do this is explained in the following sections. See the section for your operating system.

Changing the OS/2 config.sys file

Edit the config.sys file as follows:

1. Find the line that starts `SET PATH=`
2. Add to the end of the line, after a semicolon (;)
   `<drive>:\mqm\bin`
3. Find the line that starts `SET DPATH=`
4. Add to the end of the line, after a semicolon (;)
   `<drive>:\mqm\bin; <drive>:\mqm\msg`
5. Find the line that starts `SET HELP=`
6. Add to the end of the line, after a semicolon (;)
   `<drive>:\mqm\bin`
7. Find the line that starts `SET LIB=`
8. Add to the end of the line, after a semicolon (;)
   `<drive>:\mqm\lib`
9. Find the line that starts `LIBPATH=` if there is one,
10. Add to the end of the line, after a semicolon (;)
    `<drive>:\mqm\dll`
11. If there is no `LIBPATH=` line, create a new line: `LIBPATH=<drive>:\mqm\dll`

You must restart your system before these changes will take effect.

Changing the autoexec.bat file for DOS and Windows 3.1

Edit the autoexec.bat file as follows:

1. Find the line that starts `PATH`
2. Add to the end of the line, after a semicolon (;)
   `<drive>:\<dir>`
   using the drive and directory that you created on your system for the MQSeries client files.
3. If there is no `PATH` line, create a new line: `PATH <drive>:\<dir>`
4. Find the line that starts `APPEND`
5. Add to the end of the line, after a semicolon (;)
   `<drive>:\<dir>`
   using the drive and directory that you created on your system for the MQSeries client files.
6. If there is no `APPEND` line, create a new line: `APPEND <drive>:\<dir>`

You must restart your system before these changes will take effect.

| ## MQSeries Client for VM/ESA Version 2 Release 3

| The MQSeries Client for VM/ESA V2.3 is supplied as part of the VM/ESA product.
| The client is installed as a component of CMS during the installation of CMS.

| The client code resides on MAINT 193 minidisk and can be accessed by linking to
| MAINT 193.

| To set up your client and server installation and to check that the communication
| link is working, see Chapter 5, “Verifying the installation” on page 65.

Chapter 5. Verifying the installation

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

How does it work?

Instructions are given on how to use the supplied sample PUT and GET programs to verify that an MQSeries client has been installed correctly, by guiding you through the following tasks:

1. Setting up the server
2. Setting up the MQSeries client
3. Putting a message on the queue
4. Getting the message from the queue
5. Ending verification

The installation used for the example

These instructions assume that:

- The full MQSeries product has been installed on a server:
  - The Base Product and Distributed Queuing without CICS, and the Client Attachment feature on MVS/ESA.
  - The full MQSeries for AS/400 product on AS/400 platforms.
  - The *Base Product and Server* on other platforms.
- The MQSeries client software has been installed on another machine (or on the same machine), including the transfer of the MQSeries client files, where necessary.

The transmission protocol used in the example is TCP/IP. It is assumed that you have TCP/IP configured on the server and the MQSeries client machines, and that it has been initialized on both the machines. There are more details about this in Chapter 6, “Configuring communication links” on page 75.

Note: Compiled samples amqsputc and amqsgetc (amqsputw and amqsgetw for Windows 3.1) are included in the MQSeries client directories that you installed, either directly or by copying the files across as described in “Installing the MQSeries server” on page 52.

What the example shows

The following example shows how to create a queue manager called *queue.manager.1* (on platforms other than MVS/ESA), a local queue called *QUEUE1*, and a server-connection channel called *CHANNEL1* on the server. It shows how to create the client-connection channel on the MQSeries client workstation; and how to use the sample programs to put a message onto a queue, and then get the message from the queue.

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

Security

The example does ***not*** address any client security issues. See Chapter 9, “Setting up MQSeries client security” on page 117 for details if you are concerned with MQSeries client security issues.

Setting up the server (not MVS/ESA or AS/400)

Create a directory to hold working files, for example `mqverify`, and make this the current directory. Then follow the steps below to set up the server workstation.

1. Create a default queue manager (called *queue.manager.1*) by entering the following command at the command prompt:

   ```
   crtmqm -q queue.manager.1
   ```

2. Start the queue manager by entering the following command:

   ```
   strmqm
   ```

3. If you are using an MQSeries Version 5 product go on to the next step.

   If you are using an MQSeries non-Version 5 product, define the default system objects by entering the following command:

   ```
   runmqsc queue.manager.1 <PATH/amqscoma.tst >defobj.out
   ```

   where `PATH` depends on the platform you are using; see the *System Management Guide* for your platform for the value of `PATH`. When this command has completed examine the file `defobj.out` that is written to the current directory, to confirm that all the default objects were created successfully. The last line of this file should read:

   ```
   0 commands cannot be processed
   ```

   If there are commands that cannot be processed, you need to check your server installation. See the *System Management Guide* for your server platform.

4. Start MQSeries commands (MQSC) by entering the following command:

   ```
   runmqsc
   ```

   MQSC does not provide a prompt, but should respond with the message:

   ```
   Starting MQSeries Commands
   ```

5. Create a local queue by entering the following command:

   ```
   DEFINE QLOCAL(QUEUE1)
   ```

6. Create a server-connection channel by entering the following command:

   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ')
   ```

   See “Access control” on page 118 for information about `MCAUSER`.

7. Stop MQSC by pressing Ctrl+D (on MQSeries Version 5 products type `end`) and then Enter.

8. Configure the system to start channels

   On OS/2 or Windows NT

   Start a listener by entering the following command at the command prompt:

   ```
   RUNMQLSR -t tcp -m queue.manager.1
   ```

On UNIX systems
Configure the **inetd** daemon to start the MQI channels. See "TCP/IP on a UNIX system server" on page 80 for details of how to do this.

On Digital OpenVMS
See the book *MQSeries for Digital OpenVMS System Management Guide* for details of how to configure TCP/IP services to start channels.

Setting up the server (MVS/ESA)

Customize your MQSeries for MVS/ESA installation as described in the *MQSeries for MVS/ESA System Management Guide*. This includes defining the default system objects and enabling Distributed Queuing without CICS. You do not require the Batch/TSO, CICS, or IMS adapters to run as servers for MQSeries applications running on a client. However, depending on how you choose to issue commands, you may need the Batch/TSO adapter and the operations and control panels to perform administration for clients.

Now follow the steps below. You can use any of the valid command input methods to issue the MQSeries commands (MQSC) shown.

1. Start the queue manager by entering the following command:
   ```
   START QMGR
   ```
2. Create a local queue by entering the following command:
   ```
   DEFINE QLOCAL(QUEUE1)
   ```
3. Create a server-connection channel by entering the following command:
   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ')
   ```
4. Start the channel initiator by entering the following command:
   ```
   START CHINIT
   ```
5. Start the listener by entering the following command:
   ```
   START LSTR TRPTYPE(TCP) PORT(port-number)
   ```

Setting up the server (AS/400)

These instructions assume that no queue manager or other MQSeries objects have been defined. Follow these steps:

1. Create a queue manager by entering the following command:
   ```
   CRTMQM MQMNAME('qmgr')
   ```
2. Start the queue manager by entering the following command:
   ```
   STRMQM
   ```
3. Set up the default system objects
   ```
   CALL QMQM/AMQSDEF4
   ```
4. Create a local queue
   ```
   CRTMQMQ QNAME(QUEUE1)  QTYPE(*LCL)
   ```
5. Create a server-connection channel
   ```
   | CRTMQMCHL CHLNAME(CHANNEL1) CHLTYPE(*SVRCN) TRPTYPE(*TCP)
   |           MCAUSRID('QMQM')
   ```

Note: `QMQM` is the Default User ID.

6. Start the listener

```
STRMQMLSR
```

Setting up the MQSeries client

When an MQSeries application is run on the MQSeries client, the information it requires is the name of the MQI channel, the communication type, and the address of the server to be used. You provide this by defining a client-connection channel. The name used must be same as the name used for the server-connection channel defined on the server. In this example the MQSERVER environment variable is used to define the client-connection channel. This is the simplest way, although not the only one.

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

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

Defining a client-connection channel, using MQSERVER

Create a client-connection channel by setting the MQSERVER environment variable. (For more information, see Chapter 8, "Using MQSeries environment variables" on page 111).

For OS/2, DOS, Windows 3.1, Windows 95, and Windows NT clients, enter the following command:

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

For UNIX clients, enter the following command:

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

For VM/ESA clients, enter the following command:

```
GLOBALV SELECT CENV SETLP MQSERVER SYSTEM.DEF.SVRCONN/TCP/server-address(port)
```

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

Putting a message on the queue

On the MQSeries client workstation, put a message on the queue using the amqsputc sample program (amqsputw on Windows 3.1, AMQSPUT0 on VM/ESA.):

On the MQSeries client workstation (not Windows 3.1, or VM/ESA)

- From a command prompt window, change to the directory containing the sample program amqsputc.exe. This is in the `/samp/bin` directory, or the `\bin` directory for some Version 5 products. Then enter the following command:

  ```
  amqsputc QUEUE1 qmgr
  ```

  where `qmgr` is the name of the queue manager on the server (`queue.manager.1` in the non-MVS/ESA example above).
- The following message is displayed:

  ```
  Sample AMQSPUT0 start
  target qname is QUEUE1
  ```
- Type some message text and then press Enter ***twice***.
- The following message is displayed:

  ```
  Sample AMQSPUT0 end
  ```
- The message is now on the queue.

On the MQSeries client workstation (Windows 3.1)

This program has no visible interface. All messages are put in the output file, not to stdout.

This program takes four parameters, all are required:

1. The name of the output file
2. The name of the input file
3. The name of the queue manager
4. The name of the target queue

Either of these two methods can be used:

- In the DOS prompt window, enter the program name followed by the parameters. For example:

  ```
  amqsputw outfile.out infile.in qmgr QUEUE1
  ```

  Or
- To run AMQSPUTW from the Windows Program Manager, select the menu item:
  - File/Run...
  - On the Run dialog, enter the program name followed by the parameters. For example:

    ```
    amqsputw outfile.out infile.in qmgr QUEUE1
    ```

Where `outfile.out` is used to hold the messages generated when the program runs.

`infile.in` contains the data to be put onto the target queue. Each line of data is put as a message. `infile.in` must be an ASCII file.

`qmgr` is the name of the queue manager on the server (`queue.manager.1` in the non-MVS/ESA example above).

It is important ***always*** to look in the output file to see what has happened, as there is no visible indication of success or failure when you run this program.

Note: The AMQSPUTC (or AMQSPUTW) sample program starts the channel between the client and the server. When you have put the message on the queue, the sample program ends and the channel between the client and server also ends automatically.

On the MQSeries client workstation (VM/ESA)

The sample programs provided with MQSeries Client for VM/ESA V2.3 must be compiled on your system. For details see the *VM/ESA CMS Application Development Guide*.

- Enter the following command:

  ```
  AMQSPUT0 QUEUE1 qmgr
  ```

  where `qmgr` is the name of the queue manager on the server (`queue.manager.1` in the non-MVS/ESA example above).

- The following message is displayed:

  ```
  Sample AMQSPUT0 start
  target qname is QUEUE1
  ```

- Type some message text and then press Enter ***twice***.
- The following message is displayed:

  ```
  Sample AMQSPUT0 end
  ```

- The message is now on the queue.

Getting the message from the queue

On the MQSeries client workstation, get a message from the queue using the amqsgetc sample program (amqsgetw on Windows 3.1, AMQSGET0 on VM/ESA.):

On the MQSeries client workstation (not Windows 3.1, or VM/ESA)

- Change to the directory containing the sample programs, and then enter the following command:

  ```
  amqsgetc QUEUE1 qmgr
  ```

  Where `qmgr` is the name of the queue manager on the server (queue.manager.1 in the non-MVS/ESA example above).

- The message on the queue is removed from the queue and displayed.

On the MQSeries client workstation (Windows 3.1)

This program has no visible interface. All messages are put in the output file, not to stdout.

This program takes three parameters, all are required:

1. The name of the output file
2. The name of the queue manager
3. The name of the target queue

Either of these two methods can be used:

- In the DOS prompt window, enter the program name followed by the parameters. For example:

  ```
  amqsgetw outfile.out qmgr QUEUE1
  ```

  Or

- To run AMQSGETW from the Windows Program Manager, select the menu item:

  - File/Run...
  - On the Run dialog, enter the program name followed by the parameters. For example:

    ```
    amqsgetw outfile.out qmgr QUEUE1
    ```

  where:

  `outfile.out` is used to hold the messages generated when the program runs.

  `qmgr` is the name of the queue manager on the server (`queue.manager.1` in the non-MVS/ESA example above).

It is important ***always*** to look in the output file to see what has happened as there is no visible indication of success or failure when you run this program.

On the MQSeries client workstation (VM/ESA)

The sample programs provided with MQSeries Client for VM/ESA V2.3 must be compiled on your system. For details see the *VM/ESA CMS Application Development Guide*.

- Enter the following command:

  ```
  AMQSGET0 QUEUE1 qmgr
  ```

  Where `qmgr` is the name of the queue manager on the server (queue.manager.1 in the non-MVS/ESA example above).

- The message on the queue is removed from the queue and displayed.

Ending verification

The verification process is now complete.

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

```
endmqm queue.manager.1
```

or, on MVS/ESA: `STOP CHINIT` followed by `STOP QMGR`

If you want to delete the queue manager on the server (not MVS/ESA) type:

```
dltmqm queue.manager.1
```

Part 2. System administration

Chapter 6. Configuring communication links

This chapter tells you how to configure the MQSeries client and server communication links, and how to enable the server to listen for communications from the MQSeries client.

In MQSeries the logical communication links are called *channels*. You set up channel definitions at each end of your link so that your MQSeries application on the MQSeries client can communicate with the queue manager on the server. There is a detailed description of how to do this in Chapter 7, “Using channels” on page 99.

Before you define your MQI channels:

1. Decide on the form of communications you are going to use.
2. Define the connection at each end:
   - Configure the connection.
   - Record the values of the parameters that you will need for the channel definitions later on.
   - Enable the server to detect incoming network requests from your MQSeries client. This involves starting a *listener*.

This chapter explains how to do these steps.

If you already know which communication type you are using, go straight to the relevant section:

- “Defining a TCP/IP connection” on page 77
- “Defining an LU 6.2 connection” on page 82
- “Defining a NetBIOS connection” on page 91
- “Defining an SPX connection” on page 93
- “Defining a DECnet connection” on page 96

Deciding which communication type to use

There are five types of communication for MQSeries on different platforms:

- TCP/IP
- LU 6.2
- NetBIOS
- SPX
- DECnet

When you define your MQI channels, each channel definition must specify a Transmission protocol (Transport Type) attribute. A server is not restricted to one protocol, so different channel definitions can specify different protocols. For MQSeries clients, it may be useful to have alternate MQI channels using different transmission protocols.

Your choice of transmission protocol also depends on your particular combination of MQSeries client and server platforms. The possible combinations are shown in the following table.

Table 2. Transmission protocols - combination of MQSeries client and server platforms

Transmission protocol	MQSeries client	MQSeries server
TCP/IP	Digital OpenVMS DOS OS/2 UNIX systems VM/ESA Windows 3.1 Windows 95 Windows NT	AS/400 Digital OpenVMS MVS/ESA OS/2 Tandem NSK UNIX systems Windows NT
LU 6.2	Digital OpenVMS	MVS/ESA
LU 6.2	OS/2 UNIX systems VM/ESA Windows NT	AS/400 MVS/ESA OS/2 Tandem NSK UNIX systems Windows NT
NetBIOS	DOS OS/2 Windows 3.1 Windows 95 Windows NT	OS/2 Windows NT
SPX	DOS OS/2 Windows 3.1 Windows 95 Windows NT	OS/2 Windows NT
DECnet	Digital OpenVMS	Digital OpenVMS

Now go to the relevant section:

- “Defining a TCP/IP connection” on page 77
- “Defining an LU 6.2 connection” on page 82
- “Defining a NetBIOS connection” on page 91
- “Defining an SPX connection” on page 93
- “Defining a DECnet connection” on page 96

Defining a TCP/IP connection

The steps to take are detailed in the sections that follow:

On the MQSeries client
Initialize TCP/IP

On the server There are three things to do:

1. Decide on a port number.

   The port to connect to will default to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to MQSeries.

2. Initialize TCP/IP, and record the network address of the server machine.

3. Configure files (or run a command) to specify the port number and to run a listener program (non-MVS/ESA). On MVS/ESA, start a channel initiator and a listener.

Note: For more detailed step-by-step examples, see the book *MQSeries Intercommunication*.

TCP/IP on an MQSeries client (any platform)

Initialize TCP/IP.

The channel definitions that you create later will include the network address and port number of the server to which the MQSeries client is sending.

TCP/IP on an OS/2 server

First initialize TCP/IP, and record the network address of the server machine (displayed in a box as TCP/IP initializes).

Then there are two alternative methods; using INETD or using the Run Listener (RUNMQLSR) command:

Using INETD

First type:

```
SET ETC
```

This will return the path to the `ETC` subdirectory.

To use **INETD** to start MQI channels, configure files as follows, where in this case the path is taken to be `TCPIP`: Note that this is case sensitive and the entries in the two files must match.

- To TCPIP\ETC\SERVICES (or MPTN\ETC\SERVICES) add the line:

  ```
  MQSeries      1414/tcp
  ```

  where 1414, the default, is the port number required.

  Alternatively, you may want to use another port, for example, port number 1822, in which case you add the line:

  ```
  MQSeries      1822/tcp
  ```

- To TCPIP\ETC\INETD.LST, add the line:

```
    MQSeries      tcp C:\MQM\BIN\AMQCRSTA [-m QMName]
```

The part in square brackets is optional and is not required for the default queue manager. If your MQSeries for OS/2 was not installed on the C drive, replace the C: above with the correct drive letter.

It is possible to have more than one queue manager on the server machine. Add a line to each of the two files, as above, for each queue manager. For example:

```
MQSeries1     1414/tcp
MQSeries2     1415/tcp

MQSeries1     tcp C:\MQM\BIN\AMQCRSTA -m QM1
MQSeries2     tcp C:\MQM\BIN\AMQCRSTA -m QM2
```

Now stop and then start the inetd program, before continuing.

Note: There can be a maximum of five outstanding connection requests queued at a single TCP/IP port.

To avoid error messages being generated by this limitation, you can define multiple ports, as described above, with only one queue manager, or with multiple queue managers.

There is no limit on the number of clients, the limitation is on the number that connect simultaneously. The system resources will, however, limit the number that can be used. A maximum of 100 clients is a reasonable number to work with.

Using the Run Listener (RUNMQLSR) command

To run the Listener supplied with MQSeries for OS/2, that starts new MQI channels as threads, use the RUNMQLSR command. For example:

```
RUNMQLSR -t tcp [-m QMNAME] [-p 1822]
```

The square brackets indicate optional parameters:

`-m QMNAME` is not required for the default queue manager.
`-p 1822` is not required if the default port number 1414 is used.

It is possible to have more than one queue manager running on the server machine. Start a listener program for each one, on different ports. For example:

```
RUNMQLSR -t tcp
RUNMQLSR -t tcp -m QM2 -p 1415
```

Note: There can be a maximum of five outstanding connection requests queued at a single TCP/IP port.

To avoid error messages being generated by this limitation, you can define multiple ports, as described above, with only one queue manager, or with multiple queue managers.

There is no limit on the number of clients, the limitation is on the number that connect simultaneously. The system resources will, however, limit the number that can be used. A maximum of 100 clients is a reasonable number to work with.

TCP/IP on a Windows NT server

TCP/IP is initialized automatically as a service during Windows NT startup, but first define the port number as follows.

To the file c:\winnt\system32\drivers\etc\services, add the line:

```
MQSeries      1414/tcp
```

where 1414, the default, is the port number required.
(On some versions the path is c:\winnt35\system32\drivers\etc\services)

Alternatively, you may want to use another port, for example, port number 1822, in which case you add the line:

```
MQSeries      1822/tcp
```

It is possible to have more than one queue manager on the server machine. Add a line, as above, for each queue manager. For example:

```
MQSeries1     1414/tcp
MQSeries2     1415/tcp
```

Note: There can be a maximum of five outstanding connection requests queued at a single TCP/IP port.

To avoid error messages being generated by this limitation, you can define multiple ports, as described above, with only one queue manager, or with multiple queue managers.

There is no limit on the number of connected clients, the limitation is on the number that connect simultaneously.

Using the Run Listener (RUNMQLSR) command

To run the Listener supplied with MQSeries for Windows NT, that starts new MQI channels as threads, use the RUNMQLSR command. For example:

```
RUNMQLSR -t tcp [-m QMNAME] [-p 1822]
```

The square brackets indicate optional parameters:

`-m QMNAME` is not required for the default queue manager.
`-p 1822` is not required if the default port number 1414 is used.

It is possible to have more than one queue manager running on the server machine. Start a listener program for each one, on different ports. For example:

```
RUNMQLSR -t tcp
RUNMQLSR -t tcp -m QM2 -p 1415
```

TCP/IP on a UNIX system server

Note: The name of the installation directory on your UNIX system is represented here by **mqmtop**.

Configure the `inetd` daemon on the server, so that `inetd` will start the MQI channels. Log on as root and configure the following files:

1. Add a line in the /etc/services file:

   ```
   MQSeries      1414/tcp
   ```

   where 1414, the default, is the port number required. Alternatively, you may want to use a different port, for example, port number 1822 in which case add the line:

   ```
   MQSeries      1822/tcp
   ```

2. Add a line (case sensitive) in the inetd.conf file to call the program `amqcrsta`:

   ```
   MQSeries stream tcp nowait mqm /mqmtop/bin/amqcrsta amqcrsta [-m QM1]
   ```

   where `QM1` is the queue manager name, not required for the default queue manager.

3. The updates are active after you issue the following commands from the root user ID:

   On AIX:

   ```
   inetimp
   refresh -s inetd
   ```

   On other UNIX systems:

   ```
   kill -1 <process number of inetd daemon>
   ```

It is possible to have more than one queue manager on the server machine. Add a line to each of the two files, as above, for each queue manager. For example, in /etc/services:

```
MQSeries1     1414/tcp
MQSeries2     1415/tcp
```

and in the inetd.conf file:

```
MQSeries1 stream tcp nowait mqm /mqmtop/bin/amqcrsta amqcrsta -m QM1
MQSeries2 stream tcp nowait mqm /mqmtop/bin/amqcrsta amqcrsta -m QM2
```

Note: There can be a maximum of five outstanding connection requests queued at a single TCP/IP port.

To avoid error messages being generated by this limitation, you can define multiple ports, as described above, with only one queue manager, or with multiple queue managers.

There is no limit on the number of connected clients, the limitation is on the number that connect simultaneously.

TCP/IP on an AS/400 server

TCP/IP is normally initialized automatically as a service during AS/400 startup.

Use the Start Listener (STRMQMLSR) command to enable the MQSeries server to receive incoming client connections.

By default, the MQSeries for AS/400 TCP/IP listener program uses port 1414.

TCP/IP on an MVS/ESA server

Set up communications for MQSeries for MVS/ESA to use TCP/IP channels, as described in the book *MQSeries Intercommunication*. Then:

1. Start the channel initiator

   ```
   START CHINIT
   ```

2. Start the listener

   ```
   START LSTR TRPTYPE(TCP) PORT(port-number)
   ```

TCP/IP on a Digital OpenVMS server

Set up communications for MQSeries for Digital OpenVMS to use TCP/IP channels, as described in the book *MQSeries for Digital OpenVMS System Management Guide*.

See also the book *MQSeries Intercommunication* for details of how to configure TCP/IP services on Digital OpenVMS.

TCP/IP on a Tandem NSK server

Set up communications for MQSeries for Tandem NSK to use TCP/IP channels, as described in the book *MQSeries Intercommunication*. Then start the listener to enable the MQSeries server to receive incoming client connections. There are a number of ways of doing this, described in the book. For example:

```
runmqlsr -t tcp
```

By default, the MQSeries for Tandem NSK listener program uses port 1414.

See also the book *MQSeries for Tandem NonStop Kernel System Management Guide* for details of how to configure TCP/IP services on Tandem NSK.

Defining an LU 6.2 connection

The steps to take are detailed in the sections that follow:

On the MQSeries client

1. Configure SNA.
2. Set TpName and TpPath.
3. Establish a valid SNA session between the MQSeries client and server machines.

On the server

1. Start a listener, or create a listening attachment (non-MVS/ESA).
2. Start a channel initiator and a listener (MVS/ESA).

Note: For more detailed step-by-step examples, see the book *MQSeries Intercommunication*.

LU 6.2 on an OS/2 MQSeries client

First configure SNA so that an LU 6.2 conversation can be established between the MQSeries client machine and the server machine. See the book *Multiplatform APPC Configuration Guide*. This is supplied online with some MQSeries products as *APPC Configuration Guide (Red Book)* for information.

Set the Transaction Program name (TpName or TPNAME) as shown in the following table:

Table 3. Settings on the MQSeries client OS/2 system for a server platform

Server platform	**TPNAME**
OS/2	As specified in the OS/2 Run Listener command on the server, or defaulted from the OS/2 queue manager configuration file on the server.
Windows NT	As specified in the Windows NT Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows NT.
SunOS	As specified in the SunOS Run Listener command on the server.
Other UNIX systems	The same as the corresponding TpName in the side information on the remote queue manager on the server.
AS/400	The same as the compare value in the routing entry on the AS/400 system.
Tandem NSK	As specified in the channel definition on the server.
MVS/ESA	The same as the corresponding TpName in the side information on the remote queue manager on the server.

Establish a valid session between the two machines. You can specify the local LU that MQSeries for OS/2 will use, either by creating the MQSeries client configuration file (QM.INI), or as an environment variable. An entry in the configuration file takes precedence over the environment variable. For an MQSeries client on OS/2 the QM.INI file is located in directory C:\MQM.

In the QM.INI file, under the LU 6.2 section add the line:

```
LocalLU = Your_LU_Name
```

or specify the environment variable:

```
SET APPNLLU=Your_LU_Name
```

If nothing has been specified, your default LU will be used.

Find out the name of the partner LU alias, as defined in the MQSeries client machine's Communications Manager/2 profile. You will need this later, when you define the MQI channels - it is the Connection name (CONNAME).

SECURITY PROGRAM is always used when MQSeries for OS/2 attempts to establish an SNA session if a password and user ID are specified. Otherwise SECURITY NONE is used.

LU 6.2 on an OS/2 server

Start the listener program with the RUNMQLSR command, giving the TpName to listen on, or use Attach Manager in Communications Manager/2.

Using the RUNMQLSR command

Here is an example of a command to start the listener:

```
RUNMQLSR -t LU62 -n RECV [-m QMNAME]
```

where `RECV` is the TpName that is specified in the client channel definition at the MQSeries client end (as the "TpName to start on the remote, or server, side)". The last part in square brackets is optional and is not required for the default queue manager.

It is possible to have more than one queue manager running on the server machine. Assign a different TpName to each queue manager, and then start a listener program for each one. For example:

```
RUNMQLSR -t LU62 -m QM1 -n TpName1
RUNMQLSR -t LU62 -m QM2 -n TpName2
```

Using Communications Manager/2

You can use Attach Manager in Communications Manager/2 to start the listener program. You must supply a Transaction program (TP) definition specifying:

- TP name
- program path and file name
- program parameter string

You can do this using the panel configuration in Communications Manager/2, or you can edit your NDF file directly (see the heading "Define Transaction Programs" in the online *APPC Configuration Guide (Red Book)*).

Panel configuration: These are the entries required on the TP definition panel:

```
Transaction Program (TP) name  :  AMQCRS6A
OS/2 program path and file name:  c:/mqm/bin/amqcrs6a.exe
Program parameter string       :  -n AMQCRS6A
```

Note: The above is for the default queue manager, for other queue managers include `-m QMNAME` in the Program parameter string.

NDF file configuration: Your node definitions file (.ndf) must contain a `define_tp command`. The following example shows what to include:

```
define_tp
  tp_name(AMQCRS6A)
  filespec(c:/mqm/bin/amqcrs6a.exe)
  parm_string(-n AMQCRS6A)
```

Note: The above is for the default queue manager, for other queue managers include `-m QMNAME` in the parm_string.

LU 6.2 on a Windows NT MQSeries client

First configure SNA to allow an LU 6.2 conversation to be established between the MQSeries client machine and the server machine.

Set the Transaction Program name (TpName or TPNAME) as shown in the following table:

Table 4. Settings on the MQSeries client Windows NT system for a server platform

Server platform	**TPNAME**
OS/2	As specified in the OS/2 Run Listener command on the server, or defaulted from the OS/2 queue manager configuration file on the server.
Windows NT	As specified in the Windows NT Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows NT.
SunOS	As specified in the SunOS Run Listener command on the server.
Other UNIX systems	The same as the corresponding TpName in the side information on the remote queue manager on the server.
AS/400	The same as the compare value in the routing entry on the AS/400 system.
Tandem NSK	As specified in the channel definition on the server.
MVS/ESA	The same as the corresponding TpName in the side information on the remote queue manager on the server.

Create a CPI-C Side Object (symbolic destination) and record this name to use later in your channel definitions as the Connection name (CONNAME).

In the CPI-C Side Object enter the Partner LU Name at the receiving machine, the TP Name and the Mode name. For example:

```
Partner LU Name             OS2ROG2
Partner TP Name             recv
Mode Name                   #INTER
```

SECURITY PROGRAM is used, where supported by CPI-C, when MQSeries attempts to establish an SNA session.

LU 6.2 on a Windows NT server

You can use either of these methods:

- Start the listener program with the RUNMQLSR command, giving the TpName to listen on. This starts a thread to process each inbound client connection.
- Use one of the SNA servers listed (see page "Windows NT client: hardware and software required" on page 29) to set up an invokable transaction program (TP). Then invoke amqcrs6a as a separate process for each client connection. The TpName should match that specified in the CPI-C side object information referenced by CONNAME in the client-connection channel definition (see Chapter 7, "Using channels" on page 99).

Using the RUNMQLSR command

Example of the command to start the listener:

```
RUNMQLSR -t LU62 -n RECV [-m QMNAME]
```

where `RECV` is the TpName that is specified at the MQSeries client end as the "TpName to start on the remote side (server)". The last part in square brackets is optional and is not required for the default queue manager.

It is possible to have more than one queue manager running on the server machine. Assign a different TpName to each queue manager, and then start a listener program for each one. For example:

```
RUNMQLSR -t LU62 -m QM1 -n TpName1
RUNMQLSR -t LU62 -m QM2 -n TpName2
```

Using an SNA server

You can use TpSetup (from the SNA Server SDK) to define amqcrs6a as an invokable TP, or set various registry values manually.

LU 6.2 on a UNIX system MQSeries client

(For SunOS see "LU 6.2 on a SunOS MQSeries client" on page 87.)

First configure SNA so that an LU 6.2 conversation can be established between the MQSeries client machine and the server machine. See the online book *APPC Configuration Guide (Red Book)* for information, or for Sun Solaris see the book *Sunlink P2P LU 6.2 Programmer's Guide*.

Set the Transaction Program name (TpName or TPNAME) as shown in the following table:

Table 5. Settings on the MQSeries client UNIX system for a server platform

Server platform	TPNAME
OS/2	As specified in the OS/2 Run Listener command on the server, or defaulted from the OS/2 queue manager configuration file on the server.
Windows NT	As specified in the Windows NT Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows NT.
SunOS	As specified in the SunOS Run Listener command on the server.
Other UNIX systems	The same as the corresponding TpName in the side information on the remote queue manager on the server.
AS/400	The same as the compare value in the routing entry on the AS/400 system.
Tandem NSK	As specified in the channel definition on the server.
MVS/ESA	The same as the corresponding TpName in the side information on the remote queue manager on the server.

Create a CPI-C Side Object (symbolic destination) and record this name to use later in your channel definitions as the Connection name (CONNAME).

On Sun Solaris, set the environment variable APPC_LOCAL_LU to refer to the name of your Local LU.

On SINIX create a XSYMDEST entry in the TRANSIT KOGS file:

```
XSYMDEST  sendMP01,
                   RLU         = forties,
                   MODE        = MODE1,
                   TP          = recvMP01,
                   TP-TYP      = USER,
                   SEC-TYP     = NONE
```

On DC/OSx create an entry in the /etc/opt/lu62/cpic_cfg file:

```
sendMP01 <local LU name> <remote LU name> <mode name> <remote TP name>
```

On other UNIX systems, in the CPI-C Side Object enter the Partner LU Name at the receiving machine, the TpName and the Mode Name. For example:

```
Partner LU Name             OS2ROG2
Remote TP Name              recv
Service Transaction Program no
Mode Name                   #INTER
```

SECURITY PROGRAM is used, where supported by CPI-C, when MQSeries attempts to establish an SNA session.

LU 6.2 on a UNIX server

(For SunOS see “LU 6.2 on a SunOS server” on page 88.)

On the server create a TPN profile. In the TPN profile, enter the full path to the executable and the Transaction program name. For example:

```
Full path to TPN executable   /mqmtop/bin/amqcrs6a
Transaction Program name      recv
User ID                       0
```

On SINIX create an XTP entry in the TRANSIT KOGS file:

```
XTP   recvMP01,
               UID         = guenther,
               TYP         = USER,
               PATH        = /home/guenther/recvMP01.sh,
               SECURE      = NO
```

The file /home/guenther/recvMP01.sh contains:

```
#!/bin/ksh
#
# script to start the receiving side for the qmgr MP01
#
exec /opt/mqm/bin/amqcrs6a -m <queue manager>
```

You cannot use LU 6.2 for an MQSeries server running on DC/OSx because the DC/OSx SNA implementation does not support the ACCEPT verb (use TCP/IP instead).

The `User ID` field may specify a user who is a member of the mqm group.

You may require to use a queue manager other than the default queue manager. If so, define a command file that includes:

```
amqcrs6a -m Queue_Man_Name
```

and call the command file.

LU 6.2 on a SunOS MQSeries client

First configure SNA so that an LU 6.2 conversation can be established between the MQSeries client machine and the server machine. See the online book *APPC Configuration Guide (Red Book)* for information.

Set the Transaction Program name (TpName or TPNAME) as shown in the following table:

Table 6. Settings on the MQSeries client SunOS system for a server platform

Server platform	TPNAME
OS/2	As specified in the OS/2 Run Listener command on the server, or defaulted from the OS/2 queue manager configuration file on the server.
Windows NT	As specified in the Windows NT Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows NT.
SunOS	As specified in the SunOS Run Listener command on the server.
Other UNIX systems	The same as the corresponding TpName in the side information on the remote queue manager on the server.
AS/400	The same as the compare value in the routing entry on the AS/400 system.
Tandem NSK	As specified in the channel definition on the server.
MVS/ESA	The same as the corresponding TpName in the side information on the remote queue manager on the server.

Establish a valid session between the two machines.

Find out the gateway name of the client machine. You will need this later, when you define the MQI channels - it is the Connection name (CONNAME).

LU 6.2 on a SunOS server

Start the listener program with the RUNMQLSR command, giving the TpName to listen on and the gateway name of the server machine.

Here is an example of a command to start the listener:

```
RUNMQLSR -t LU62 -n RECV -g GNAME1 [-m QMNAME]
```

where `RECV` is the TpName that is specified at the MQSeries client end as the “TpName to start on the remote side (server)”, and `GNAME1` is the gateway name of the server machine. The last part in square brackets is optional and is not required for the default queue manager.

It is possible to have more than one queue manager running on the server machine. Assign a different TpName to each queue manager, and then start a listener program for each one. For example:

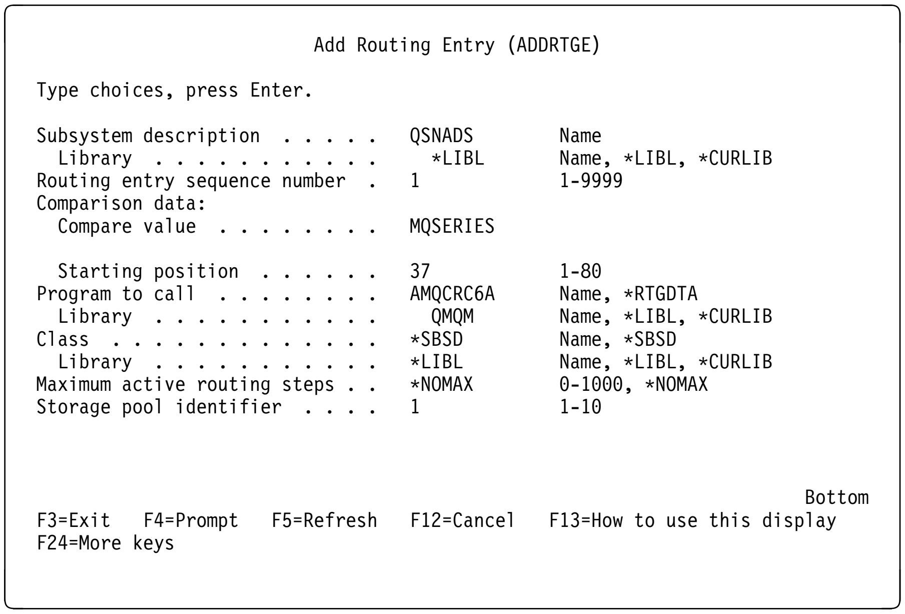
```
RUNMQLSR -t LU62 -m QM1 -n TpName1 -g GNAME1
RUNMQLSR -t LU62 -m QM2 -n TpName2 -g GNAME1
```

LU 6.2 on an AS/400 server

Use ADDRTGE command to add a routing entry to a subsystem at the initiated end to enable the initiating end to start the channel. The routing entry specifies the program that is invoked when the channel starts.

Alternatively, create and start a new subsystem by using the Work with Subsystem Descriptions (WRKSBSD) panel.

The ADDRTGE panel is shown in Figure 1.



```
                         Add Routing Entry (ADDRTGE)

 Type choices, press Enter.

 Subsystem description  . . . . .   QSNADS        Name
   Library  . . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
 Routing entry sequence number  .   1             1-9999
 Comparison data:
   Compare value  . . . . . . . .   MQSERIES

   Starting position  . . . . . .   37            1-80
 Program to call  . . . . . . . .   AMQCRC6A      Name, *RTGDTA
   Library  . . . . . . . . . . .     QMQM        Name, *LIBL, *CURLIB
 Class  . . . . . . . . . . . . .   *SBSD         Name, *SBSD
   Library  . . . . . . . . . . .   *LIBL         Name, *LIBL, *CURLIB
 Maximum active routing steps . .   *NOMAX        0-1000, *NOMAX
 Storage pool identifier  . . . .   1             1-10



                                                                    Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 1. LU 6.2 communication setup panel - initiated end

Subsystem description
The name of your subsystem where this definition resides. Use the AS/400 WRKSBSD command to view and update the appropriate subsystem description for the routing entry.

Routing entry sequence number
A unique number in your subsystem to identify this communication definition. It may be set to a number from 1 to 9999.

Comparison data: compare value
A text string to compare with that received when the session is started by **transaction program** parameter. The value can be any unique string. The character string is derived from the Transaction program field of the sender CSI.

Comparison data: starting position
The character position in the string where the comparison is to start.

Note: The starting position field is the character position in the string for comparison, and this is always 37.

Program to call	The name of program that runs the inbound message program to be called to start the session. **Note:** AMQCRC6A is a program supplied with MQSeries for AS/400 that sets up the environment and then calls AMQCRS6A.
Class	The name and library of the class used for the steps started through this routing entry. The class defines the attributes of the routing step's running environment and specifies the job priority. Specify an appropriate class entry. Use, for example, the WRKCLS command to display existing classes or to create a new class. Further information on managing work requests from remote LU 6.2 systems is available in the *AS/400 Programming: Work Management Guide*.

LU 6.2 on a Tandem NSK server

Set up communications for MQSeries for Tandem NSK to use LU 6.2 channels, as described in the book *MQSeries Intercommunication* and in the book *MQSeries for Tandem NonStop Kernel System Management Guide*.

Make sure you have an AUTOSTART(ENABLED) channel with an LU 6.2 responder process running if you are using SNAX or ICE.

LU 6.2 on an MVS/ESA server

Set up communications for MQSeries for MVS/ESA to use LU 6.2 channels, as described in the book *MQSeries Intercommunication*. Then:

1. Start the channel initiator

   ```
   START CHINIT
   ```

2. Start the listener

   ```
   START LSTR TRPTYPE(LU62) LUNAME(lu-name)
   ```

LU 6.2 on a Digital OpenVMS client

On Digital OpenVMS, SNA LU 6.2 supports only PU 2.0. Therefore communication can be to PU 5.0 only on an MVS/ESA server. If you want to communicate with a server on a platform other than MVS/ESA, you must use another protocol. See Table 2 on page 76.

First configure SNA so that an LU 6.2 conversation can be established between the MQSeries client machine and the server machine. See the book *MQSeries Intercommunication* for information.

Set the Transaction Program name (TpName or TPNAME) to the same as the corresponding TpName in the side information on the remote queue manager on the MVS/ESA server.

Establish a valid session between the two machines.

Defining a NetBIOS connection

The steps to take are detailed in the sections that follow:

On the client
Define a local NetBIOS name for the client.

On the server

1. Define a local NetBIOS name for the server.
2. Start a listener program.

NetBIOS on an MQSeries client (any suitable platform)

Define a local NetBIOS name for the client.

The local NetBIOS name that the MQSeries processes use can be specified in two ways on the MQSeries client. In order of precedence they are:

1. The `MQNAME` environment variable

   ```
   SET MQNAME=Your_env_Name
   ```

2. A queue manager configuration file (QM.INI) parameter. The QM.INI file is located as follows:

   OS/2, Windows 95, Windows NT
   `<drive>:\<dir>\mqm`
   DOS, Windows 3.1, WIN-OS/2
   The root directory of the drive where the MQSeries client is installed, or where the MQDATA environment variable points.

   The entry in QM.INI is:

   ```
   NETBIOS:
     LocalName = Your_env_Name
   ```

Note: For use with Novell NetBIOS emulation, or with NetBIOS on Windows NT, each MQSeries process should use a different local NetBIOS name.

NetBIOS on an OS/2 server

Define a local NetBIOS name for the server.

The local NetBIOS name that the MQSeries processes use can be specified in three ways on the server. In order of precedence they are:

1. The `-l` parameter on the RUNMQLSR command
2. The `MQNAME` environment variable

   ```
   SET MQNAME=Your_env_Name
   ```

3. A queue manager configuration file (QM.INI) parameter. QM.INI is located in `\mqm\qmgrs\QueueManagerName`

   The entry in QM.INI is:

   ```
   NETBIOS:
     LocalName = Your_env_Name
   ```

Note: For use with Novell NetBIOS emulation each MQSeries process should use a different local NetBIOS name.

Start a listener program

Start the listener program with the RUNMQLSR command, optionally giving the local NetBIOS name (`LOCALNAME`) to listen on. For example:

```
RUNMQLSR -t netbios [-m QMNAME] [-s Sessions]
[-e NAMES] [-o COMMANDS] [-l LOCALNAME]
```

See the *MQSeries Command Reference* for details of the options on the RUNMQLSR command.

Note: Both sending end (MQSeries client) and receiving end (server) ***must*** have a local NetBIOS name defined. You are strongly advised to make sure that all NetBIOS names used are unique in the network. If this is not done, unpredictable results may occur.

NetBIOS on a Windows NT server

Define a local NetBIOS name for the server.

The local NetBIOS name that the MQSeries processes use can be specified in three ways on the server. In order of precedence they are:

1. The `-l` parameter on the RUNMQLSR command. This defines the NetBIOS station name to be used with that instance of RUNMQLSR.
2. The `MQNAME` environment variable

   ```
   SET MQNAME=Your_env_Name
   ```

3. A queue manager configuration file (QM.INI) parameter. QM.INI is located in `\mqm\qmgrs\QueueManagerName`

   The entry in QM.INI is:

   ```
   NETBIOS:
     LocalName = Your_env_Name
   ```

Note: For use with NetBIOS on Windows NT each MQSeries process should use a different local NetBIOS name; if you are running multiple MQSeries applications simultaneously on the MQSeries client, they must use different NetBIOS names set using the environment variable MQNAME.

Start a listener program

Start the listener program with the RUNMQLSR command, optionally giving the local NetBIOS name (`LOCALNAME`) to listen on. For example:

```
RUNMQLSR -t netbios [-m QMNAME] [-s Sessions]
[-e NAMES] [-o COMMANDS] [-l LOCALNAME]
```

See the *MQSeries Command Reference* for details of the options on the RUNMQLSR command.

Note: Both sending end (MQSeries client) and receiving end (server) **must** have a local NetBIOS name defined.
You are strongly advised to make sure that all NetBIOS names used are unique in the network. If this is not done, unpredictable results may occur.

Defining an SPX connection

The steps to take are detailed in the sections that follow:

On the client
No action is required until you create the channel definitions.

On the server

1. Decide on a socket number.

   The socket to connect to defaults to 5E86. The default can be changed by adding a socket parameter to the qm.ini file (see the book *MQSeries Intercommunication*).

2. Determine the IPX/SPX network address and the node (LAN adapter address) of the server machine.

3. Start a listener, specifying the chosen socket.

SPX on an MQSeries client (any suitable platform)

There is no action required now other than to make sure that SPX is running on your client machine.

The channel definitions that you create later will include the SPX network address, node address, and socket number of the server to which the MQSeries client is sending.

SPX on an MQSeries server (OS/2 or Windows NT)

Find out and record the IPX/SPX network address of the server. Your network administrator has this information.

Record the node (LAN adaptor address) of the server machine.

Using the Run Listener (RUNMQLSR) command

To run the listener supplied with MQSeries, which starts new MQI channels as threads, use the RUNMQLSR command. For example:

```
RUNMQLSR -t spx [-m QMNAME] [-x 5E87]
```

The square brackets indicate optional parameters:

`-m QMNAME` is not required for the default queue manager.
`-x 5E87` is not required if the default socket number 5E86 is used.

It is possible to have more than one queue manager running on the server machine. Start a listener program for each one, on different socket numbers. For example:

```
RUNMQLSR -t spx
RUNMQLSR -t spx -m QM2 -x 5E87
```

SPX and IPX parameters

You may need to modify some of the default SPX or IPX parameters of your environment to tune its use for MQSeries. In most cases the default settings will be fine. The actual parameters, and the method of changing them vary according to your platform and the provider of the SPX communications support. The following describes some of these parameters, particularly where they could influence the operation of MQSeries channels and client connections.

SPX on an OS/2 client

Please refer to the Novell Client for OS/2 documentation for full details of the use and setting of NET.CFG parameters.

The following IPX/SPX parameters can be added to the Novell NET.CFG file, and can affect MQSeries SPX channels and client connections.

SPX

- sessions (default 16)

  This specifies the total number of simultaneous SPX connections. Each MQSeries channel, or client connection uses one session. You may need to increase this value depending on the number of MQSeries channels or client connections you need to run.

- retry count (default = 12)

  This controls the number of times an SPX session will resend unacknowledged packets. MQSeries does not override this value.

- verify timeout, listen timeout, and abort timeout (milliseconds)

  These timeouts adjust the Keepalive behaviour. If an SPX sending end does not receive anything within the verify timeout, it sends a packet to the receiving end. It then waits for the listen timeout for a response. If one is still not received, another packet is sent and a response is expected within the abort timeout period.

IPX

- sockets (range = 9 - 128, default 64)

  This specifies the total number of IPX sockets available. MQSeries channels use this resource, so depending on the number of channels and the requirements of other IPX/SPX applications, you may need to increase this value.

SPX on a DOS or Windows 3.1 client

Please refer to the Novell Client for DOS and Windows documentation, for full details of the use and setting of NET.CFG parameters.

The following IPX/SPX parameters can be added to the Novell NET.CFG file, and can affect MQSeries SPX channels and client connections.

SPX

- connections (default 15)

  This specifies the total number of simultaneous SPX connections. Each MQSeries channel, or client connection uses one session. You may need to increase this value depending on the number of MQSeries channels or client connections you need to run.

IPX

- sockets (default = 20)

  This specifies the total number of IPX sockets available. MQSeries channels use this resource, so depending on the number of channels and the requirements of other IPX/SPX applications, you may need to increase this value.

- retry count

  This controls the number of times unacknowledged packets will be resent. MQSeries does not override this value.

SPX on a Windows NT client

Please refer to the Microsoft documentation for full details of the use and setting of the NWLink SPX and IPX parameters. The IPX/SPX paramters are in the following paths in the registry:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkSPX\Parameters
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkIPX\Parameters
```

SPX on a Windows 95 client

Please refer to the Microsoft documentation for full details of the use and setting of the SPX and IPX parameters. They are accessed by selecting Network option in the control panel, then double clicking on "IPX/SPX Compatible Transport".

Defining a DECnet connection

The steps to take are detailed in the sections that follow:

On the client
No action is required until you create the channel definitions

On the server

1. Decide on an object number or task name
2. Determine the nodename of the server machine
3. Configure a DECnet object

DECnet on an MQSeries client

There is no action required now other than to make sure that DECnet is running on your client machine.

The channel definitions that you create later will include the DECnet nodename and object number, or task name, of the server to which the MQSeries client is sending.

DECnet on an MQSeries server (Digital OpenVMS)

Decide on an object number or task name.

Find out and record the nodename of the server. Your network administrator has this information.

Receiving on DECnet Phase IV

To use DECnet Phase IV to start channels, you must configure a DECnet object as follows:

1. Create a file which has the DCL command to start the DECnet receiver program, amqcrsta.exe. Place this file in the SYS$MANAGER directory as follows:

   ```
   $ create sys$manager:mqrecvdecnet.com
   $ mcr amqcrsta.exe [-m Queue_Man_Name] -t DECnet
    Ctrl-Z
   $
   ```

   Note: If you have multiple queue managers you **must** make a new file and DECnet object for each queue manager.

2. Create a DECnet object to start the receiving channel program automatically:

   ```
   $ MCR NCP
   NCP> define object MQSERIES
   Object number               (0-255): 0
   File name                (filename):sys$manager:mqrecvdecnet.com
   Privileges (List of VMS privileges):
   Outgoing connect privileges (List of VMS privileges):
   User ID           (1-39 characters): mqm
   Password          (1-39 characters): mqseries
                  (note: you must supply the correct password for MQM)
   Account           (1-39 characters):
   Proxy access (INCOMING, OUTGOING, BOTH, NONE, REQUIRED):
   NCP> set known objects all
   NCP> exit
   ```

Note: The preceding example should be amended to use proxy user identifiers rather than actual user identifiers. This will prevent any unauthorized access to the database. Information on how to set up proxy identifiers is given in the *Digital DECnet for OpenVMS Networking Manual*.

3. Ensure that all known objects are set when DECnet is started.

Receiving on DECnet OSI

Configure for MQSeries channel objects:

1. Start the NCL configuration interface:

   ```
   $ MC NCL
   NCL>
   ```

2. Create a session control application entity:

   ```
   NCL> create session control application MARK
   NCL> set sess con app MARK address {name=MARK{
   NCL> set sess con app MARK image name -
   _ SYS$SPECIFIC:[MQS_SERVER]MARK.COM
   NCL> set sess con app MARK user name "MQM"
   NCL> set sess con app MARK node synonym true

   NCL> show sess con app MARK all [characteristics]
   ```

   Note that user-defined values are in uppercase.

3. Create the com file (MARK.COM above) as for DECnet Phase IV.

4. The log file for the object is net$server.log in the sys$login dir for the user name specified for the application.

5. The MQSC configuration for conname, as for DECnet PhaseIV, is:

   ```
   NODE(APP_OR_OBJ_NAME)
   ```

Chapter 7. Using channels

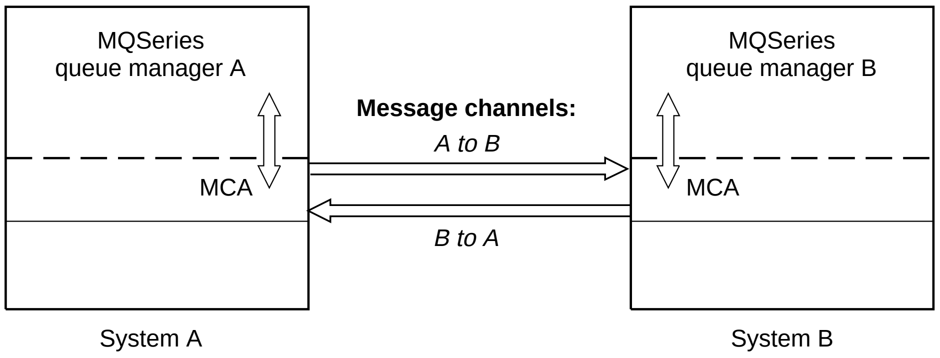
This chapter begins with a description of what *channels* are and how they are defined in an MQSeries client and server environment. Then it gives the steps you can follow to define your channels.

Some information is given about migrating from previous versions of MQSeries for OS/2 and AIX.

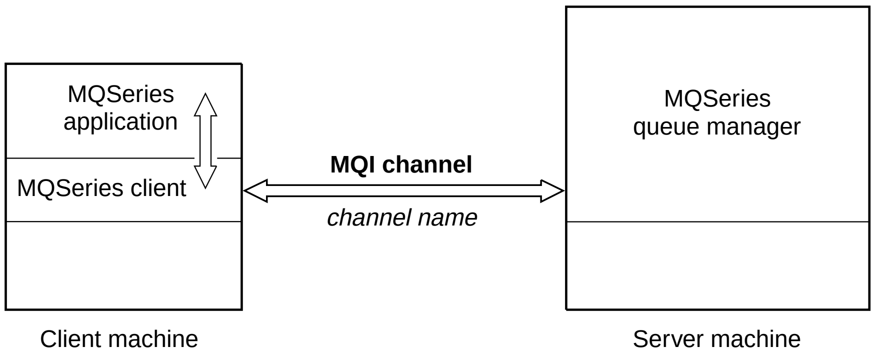
What is a channel?

A channel is a logical communication link. There are two different categories of channel in MQSeries, with different channel types within these categories:

Message channel
This connects two queue managers via *message channel agents* (MCAs), and is unidirectional. Its purpose is to transfer messages from one queue manager to another. A channel definition exists at the sending end of the link and another at the receiving end.



MQI channel
This connects an MQSeries client to a queue manager on a server machine. It is bidirectional and is for the transfer of MQI calls and responses only (including MQPUT calls that contain message data). A channel definition exists for each end of the link, and there are different ways of creating and using the channel definitions (see “Connecting the MQSeries client and server - channel definitions” on page 100).



Channel definitions, of both categories described above, must include a *channel type* as well as a *channel name*. You can choose to use different channel types according to the application you are designing, but the same channel name must be used at both ends of each combination.

Message channel types

These are not required in the MQSeries client and server environment. See the book *MQSeries Intercommunication* for details.

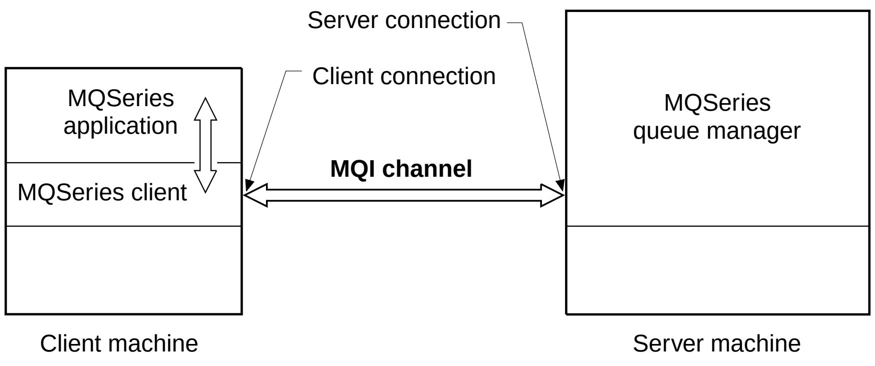
MQI channel types

There are two channel types for MQI channel definitions. They define the bi-directional MQI channel.

Client connection	This type is for the MQSeries client.
Server connection	This type is for the server running the queue manager, with which the MQSeries application, running in an MQSeries client environment, will communicate.

Connecting the MQSeries client and server - channel definitions

The connection between the MQSeries client and the queue manager on the server is a bi-directional MQI channel that is established when you issue an MQCONN or MQCONNX call. To create any new channel, you have to create **two** channel definitions, one for each end of the connection, using the same channel name and compatible channel types. In this case the channel types are *server connection* and *client connection*.



There are two different ways of creating the channel definitions and giving the MQSeries application on the MQSeries client machine access to the channel.

These two methods are described in detail in this chapter:

1. Create one channel definition on the MQSeries client and the other on the server.

   This is the easier of the two methods, and applies to any combination of MQSeries client and server platforms. Use it when you are getting started on the system, or to test your set-up.

   Create the *server connection* channel on the server machine, then use the environment variable MQSERVER on the MQSeries client machine to define a simple *client connection* channel (see Chapter 8, “Using MQSeries environment variables” on page 111).

2. Create both channel definitions on the server machine.

   Use this method when you are setting up a number of channels and MQSeries client machines at the same time.

   Note for AS/400 users: MQSeries for AS/400 does not support this method. You can use an MQSeries server on a different platform to set up the channels, or you can use the first method above.

   You can use the environment variables MQCHLLIB and MQCHLTAB on the MQSeries client machine to access the MQSeries *client channel definition table* (see “Client channel definition table” on page 106 and Chapter 8, “Using MQSeries environment variables” on page 111).

Defining your channels

First start the queue manager on the server.

Then go to the section that describes the method you are going to use:

- “Creating one definition on the MQSeries client and the other on the server” on page 102
- “Creating both definitions on the server” on page 105

Automatic definition of channels by servers (V5 and AS/400)

| The MQSeries Version 5 products and MQSeries for AS/400 V4R2 include a feature that can automatically create a channel definition on the server if one does not exist.

If an inbound attach request is received from a client and an appropriate server-connection definition cannot be found in the channel definition table, MQSeries creates a definition automatically and adds it to the channel definition table. Automatic definitions are based on two default definitions supplied with MQSeries: SYSTEM.AUTO.RECEIVER and SYSTEM.AUTO.SVRCONN. You enable automatic definition of server-connection definitions by updating the queue manager object using MQSC ALTER QMGR (or the PCF command Change Queue Manager).

For more information about the automatic creation of channel definitions, see the book *MQSeries Intercommunication*.

Creating one definition on the MQSeries client and the other on the server

Use MQSeries commands (MQSC) to define the server connection channel on the server, as follows. On MQSeries for AS/400 you can use MQSC and the CL commands. You are limited to defining one simple channel on the MQSeries client because MQSC is not available on a machine where MQSeries has been installed as an MQSeries client only.

On the server

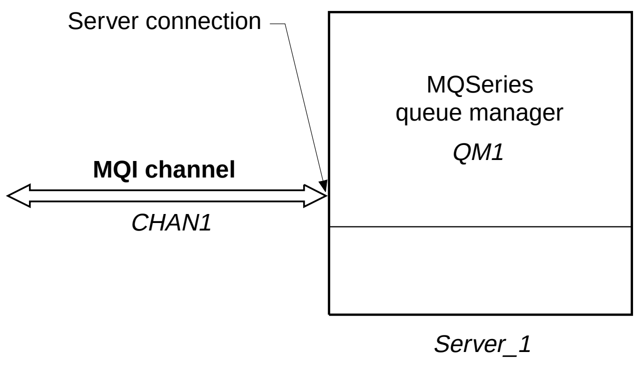
If your server platform is not MVS/ESA, you first create and start a queue manager and then start MQSeries commands (MQSC). See “Creating a queue manager and starting MQSC on the server” on page 108.

Define a channel

Define a channel with your chosen name and a channel type of *server connection*. This channel definition is kept in the *client channel definition table* associated with the queue manager running on the server (for details see “Client channel definition table” on page 106).

For example:

```
DEFINE CHANNEL(CHAN1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to Client_1')
```



On the MQSeries client

You cannot use MQSC on the MQSeries client. However, when you require a simple channel definition, without specifying all the attributes, you can use a single environment variable, MQSERVER (see Chapter 8, “Using MQSeries environment variables” on page 111).

A simple channel may be defined on OS/2, DOS, Windows 95, or Windows 3.1 as follows:

```
SET MQSERVER=ChannelName/TransportType/ConnectionName
```

Or, for OS/2 using LU 6.2 only, as follows:

```
SET MQSERVER=ChannelName/LU62/ConnectionName/ModeName/TpName
```

A simple channel may be defined on UNIX systems as follows:

```
export MQSERVER=ChannelName/TransportType/ConnectionName
```

ChannelName must be the ***same*** name as defined on the server.

TransportType may be one of the following, depending on your MQSeries client platform (see Table 1 on page 12):

LU62	For LU 6.2
TCP	For TCP/IP
NETBIOS	For NetBIOS
SPX	For SPX
DECNET	For DECnet

The ConnectionName is the name of the server machine as defined to the communications protocol (TransportType).

Note: On UNIX systems the TransportType is case sensitive and must be in uppercase. An MQCONN or MQCONNX call will return 2058 if the TransportType is not recognized

For OS/2, ModeName is the LU 6.2 mode name and TpName is the transaction program name.

Note: #INTER should be the ModeName of choice for most occasions. You can also specify Modename and TpName in your Communications Manager/2 profile. ModeName and TpName are fully described in the book *MQSeries Intercommunication*.

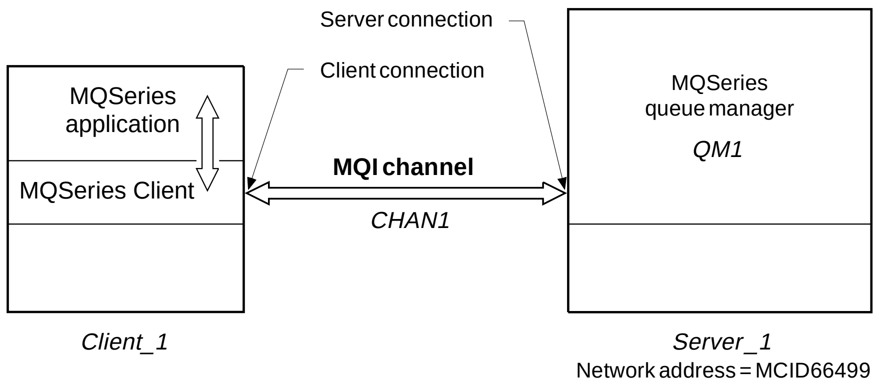
For example, on OS/2:

```
SET MQSERVER=CHAN1/TCP/MCID66499
```

or, on a UNIX system:

```
export MQSERVER=CHAN1/TCP/MCID66499
```

Note: To change the TCP/IP port number, see “MQSERVER” on page 114.



Some more examples of defining the simple channel on OS/2, DOS, Windows 3.1, Windows 95, and Windows NT are:

```
SET MQSERVER=CHAN1/TCP/9.20.4.56
SET MQSERVER=CHAN1/NETBIOS/BOX643
```

Some examples of defining the simple channel on a UNIX system are:

```
export MQSERVER=CHAN1/TCP/9.20.4.56
export MQSERVER=CHAN1/LU62/BOX99/MODENAME1/TPNAME1
```

Where `BOX99` is the LU 6.2 ConnectionName.

On the MQSeries client all MQCONN or MQCONNX requests then attempt to use the channel you have defined.

Note: The MQSERVER environment variable takes priority over any client channel definition pointed to by MQCHLLIB and MQCHLTAB.

Cancelling MQSERVER: To nullify MQSERVER and return to the client channel definition table pointed to by MQCHLLIB and MQCHLTAB, enter, on OS/2, DOS, or Windows NT:

```
SET MQSERVER=
```

or, on a UNIX system:

```
unset MQSERVER
```

Creating both definitions on the server

On the server machine use MQSeries commands (MQSC) to define the channel. For more details about the MQSC, refer to the *MQSeries Command Reference*.

Note for AS/400 users: This method cannot be used on MQSeries for AS/400.

On the server

Define the server connection and then define the client connection.

If your server platform is not MVS/ESA, you first create and start a queue manager and then start MQSeries commands (MQSC). See “Creating a queue manager and starting MQSC on the server” on page 108.

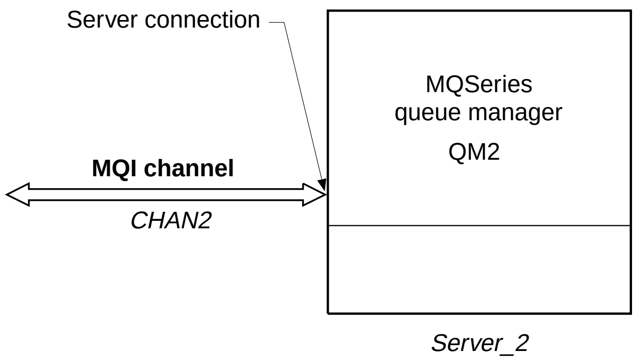
Defining the server connection

On the server machine, define a channel with your chosen name and a channel type of *server connection*.

For example:

```
DEFINE CHANNEL(CHAN2) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to Client_2')
```

This channel definition is kept in the channel definition table associated with the queue manager running on the server. The channel definition table cannot be created or updated manually. The MQSeries commands must be used as described here.



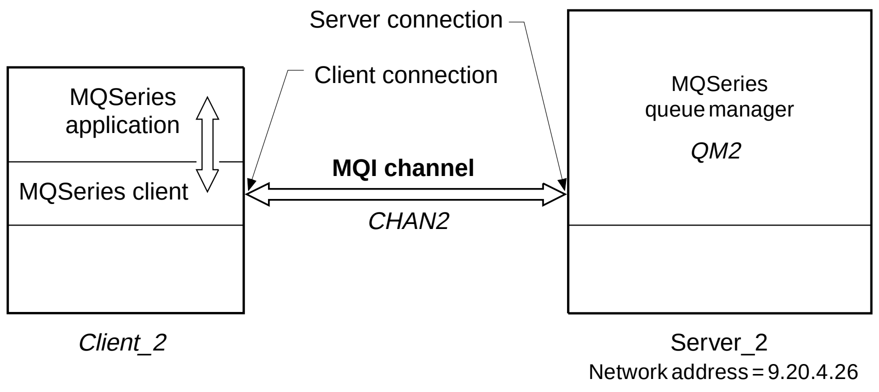
Defining the client connection

Also on the server machine, define a channel with the ***same*** name and a channel type of *client connection*.

You must state the connection name (CONNAME). For TCP/IP this is the network address of the server machine. It is a good idea to specify the queue manager name (QMNAME) to which you want your MQSeries application, running in the client environment, to connect. See Chapter 12, “Running applications on MQSeries clients” on page 135.

For example:

```
DEFINE CHANNEL(CHAN2) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) QMNAME(QM2) DESCR('Client connection to Server_2')
```



Client channel definition table

For non-MVS/ESA systems the channel definition described above is kept in the *client channel definition table* associated with the queue manager running on the server. This table is called AMQCLCHL.TAB and it is a binary file that cannot be edited directly. You use `DEFINE CHANNEL` to add entries, or `ALTER CHANNEL` to alter the attributes of a channel already in the client channel definition table.

AMQCLCHL.TAB is in the directory:

For OS/2 and Windows NT
`\mqm\qmgrs\queuemanagername\@ipcc`

For Tandem NSK
`$volume.QMD.CCHDEFS`

Where `QMD` is the data subvolume of your queue manager.

For UNIX systems
/**mqmtop**/qmgrs/QUEUEMANAGERNAME/@ipcc

Note that `QUEUEMANAGERNAME` is case sensitive for UNIX systems.

For MVS/ESA systems
Kept with all other object definitions on pageset zero.

Note for TandemNSK users: A conversion utility (CNVCLCHL) is provided to convert the client channel definition table from a Tandem structured file to an unstructured one. See the book *MQSeries for Tandem NonStop Kernel System Management Guide* for more details.

Note for VM/ESA users: If you are connecting to an MQSeries client on a VM/ESA system there is a limit to the number of client connection channels you can define. The maximum number that can be held in the client channel definition table on your server, for a VM/ESA client is 18.

Do not delete AMQCLCHL.TAB: It contains default channel definitions that are required when you define a channel. If you suspect that this has been deleted, for example you get error messages when you try to define a new channel, check to see that the file exists. If it has been deleted, define the default system objects again as described in “Setting up the server (not MVS/ESA or AS/400)” on page 66.

On the MQSeries client

On the MQSeries client machine, use the environment variables MQCHLLIB and MQCHLTAB to allow the MQSeries application to access the client channel definition table on the server (but not a server on AS/400 or MVS/ESA).

MQCHLLIB Specifies the path to the directory containing the channel definition table. If not specified, the default used is `DefaultPrefix` from the mqs.ini file.

Note: The channel definition table is not automatically created in the `DefaultPrefix` directory. If you do not specify the MQCHLLIB environment variable, copy to the `DefaultPrefix` directory, the channel definition table that you want the client to use.

MQCHLTAB Specifies the name of the file to use. If not specified, the default client channel definition table name (AMQCLCHL.TAB) is used.

For example, to set the environment variables on a UNIX system, type:

```
export MQCHLLIB=/var/mqm/qmgrs/QUEUEMANAGERNAME/@ipcc
export MQCHLTAB=AMQCLCHL.TAB
```

AMQCLCHL.TAB is the default.

In many cases the MQCHLLIB and MQCHLTAB variables might be used to point to a client channel definition table on a file server that is used by many MQSeries clients.

Alternatively, or if this is not possible, you can copy the client channel definition table, AMQCLCHL.TAB (a binary file), onto the client machine and again use MQCHLLIB and MQCHLTAB to specify where the client channel definition table is.

On MVS/ESA, use the COMMAND function of the CSQUTIL utility to make a client channel definition file that can then be downloaded to the client machine using a file-transfer program. For details see the book *MQSeries for MVS/ESA System Management Guide*.

For example:

```
//CLIENT    EXEC  PGM=CSQUTIL,PARM='QM2'
//STEPLIB   DD   DISP=SHR,DSN=thlqual.SCSQANLE
//          DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTCLNT   DD   DISP=OLD,DSN=MY.MQSERIES.CLIENTS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
COMMAND DDNAME(CMDCHL) MAKECLNT(OUTCLNT) CCSID(437)
/*
//CMDCHL    DD *
DISPLAY CHANNEL(*) ALL TYPE(CLNTCONN)
/*
```

where `thlqual` is a high level qualifier for the MQSeries library data sets. The data set for the client channel definition file (specified by DDname `OUTCLNT` in the example) must have the format:

```
RECFM=U, LRECL=2048, BLKSIZE=2048
```

If you use FTP to copy the file, remember to type `bin` to set binary mode; do not use the default ascii mode.

Notes:

1. The MQCHLLIB and MQCHLTAB environment variables are honored by the MQSeries commands when defining client connection channels. Therefore, for client connection channels only, you can use the MQCHLLIB and MQCHLTAB environment variables to override the default name and location of the generated client channel definition table.
2. The client channel definition pointed to by MQCHLLIB and MQCHLTAB may be overridden by the MQSERVER environment variable.

Migrating from MQSeries for OS/2 V2.0 and MQSeries for AIX V2.1 or V2.2

The internal format of the channel definition table has been changed in the MQSeries for OS/2 Version 2.0.1 and MQSeries for AIX Version 2.2.1 products.

For MQSeries clients, you may have to re-create your client channel definition table (by default, this is called AMQCLCHL.TAB). Earlier Version 2 clients can still access your client channel definition table, provided none of the MQSeries client definitions are changed by a queue manager created under MQSeries for OS/2 Version 2.0.1 or MQSeries for AIX Version 2.2.1.

If they are changed by a queue manager from one of these products, an MQSeries client from previous levels of MQSeries for OS/2 or AIX will no longer be able to read the file. You will receive a MQRC_Q_MGR_NOT_AVAILABLE return code from your application, with an error message in the error log file of AMQ9517 - File damaged. In this case, reinstall the MQSeries client code from an MQSeries for OS/2 Version 2.0.1 or MQSeries for AIX Version 2.2.1 server machine. In the case of a DOS MQSeries client, also relink your application.

Creating a queue manager and starting MQSC on the server

First create a queue manager, called QM1 for example:

```
crtmqm QM1
```

Start the queue manager:

```
strmqm QM1
```

If you are using one of the MQSeries Version 5 products, the default objects are defined automatically when you create the queue manager. For other MQSeries products, define the default objects as explained in “Setting up the server (not MVS/ESA or AS/400)” on page 66.

Start MQSeries commands (MQSC)

| On all platforms except AS/400, start MQSC by entering the following command:

```
runmqsc QM1
```

| On AS/400 use `STRMQMMQSC` and specify which file or member contains the runmqsc
| statements.

MQSC does not provide a prompt, but should respond with the message:

```
Starting MQSeries Commands
```

Chapter 8. Using MQSeries environment variables

This chapter describes the environment variables that you can use with MQI applications. They are available on all the MQSeries client platforms unless otherwise stated.

Note for AS/400 and MVS/ESA users: MQSeries for AS/400 and MQSeries for MVS/ESA users'. do not support ***any*** MQSeries environment variables. If you are using either of these platforms as your server, see “MQCHLLIB” on page 112 for information about the location of the client channel definition table. You can still use the MQSeries environment variables on your client platform.

| **Note for Tandem NSK users:** MQSeries for Tandem NSK does not support ***any***
| MQSeries environment variables. MQSeries for Tandem NSK does recognize
| TACL environment variables, or PARAMS. See the *MQSeries for Tandem*
| *NonStop Kernel System Management Guide* for details. You can still use the
| MQSeries environment variables on your client platform.

The MQSeries environment variables are:

- MQCCSID
- MQCHLLIB
- MQCHLTAB
- MQDATA (DOS, Windows 3.1, and Windows NT only)
- MQNAME
- MQ_PASSWORD (see Chapter 9)
- MQSERVER
- MQSPREFIX (used on the server, see the *MQSeries System Administration* book)
- MQTRACE (DOS, Windows 3.1, and VM/ESA only)
- MQ_USER_ID (see Chapter 9)
- MQSWORKPATH (OS/2 only)

MQSeries uses default values for those variables that you have not set. Update your system profile to make a permanent change; issue the command from the command line to make a change for this session only, or if you want one or more variables to have a particular value dependent on the application that is running, add commands to a command script file used by the application.

For each environment variable, use the command relevant to your platform to display the current setting or to reset the value of a variable.

For example:

Command	Effect
SET MQSERVER=	Removes the variable from OS/2, DOS, Windows 3.1, Windows 95, and Windows NT environments
unset MQSERVER	Removes the variable from UNIX systems environments
SET MQSERVER	Displays the current setting on OS/2, DOS, Windows 3.1, and Windows NT
echo $MQSERVER	Displays the current setting on UNIX systems
set	Displays all environment variables for the session

MQCCSID

This specifies the coded character set number to be used and overrides the machine's configured CCSID.

The format of this command is:

For OS/2 and Windows NT
  SET MQCCSID=number

For UNIX systems
  export MQCCSID=number

| For VM/ESA
| GLOBALV SELECT CENV SETLP MQCCSID number

MQCHLLIB

This holds the path to the directory containing the client channel definition table, on the MQSeries client. If MQCHLLIB is not set, the path defaults to:

For OS/2, DOS, Windows 3.1, Windows 95, and Windows NT
  Rootdrive:\mqm\

For UNIX systems
  /var/mqm/

If you are using MQSeries for AS/400 or MVS/ESA as your server, the file must be kept on the MQSeries client machine. For servers on other platforms, consider keeping this file on the server, to make administration easier.

The format of this command is:

For OS/2, DOS, Windows 3.1, Windows 95 and Windows NT
  SET MQCHLLIB=pathname

For UNIX systems
  export MQCHLLIB=pathname

For example:

```
SET MQCHLLIB=C:\os2
```

MQCHLTAB

This specifies the name of the client channel definition table. The default file name is `amqclchl.tab`. This is found on the server machine, in the directory:

For OS/2, DOS, Windows 3.1, Windows 95, and Windows NT
\mqm\qmgrs\queuemanagername\@ipcc

For UNIX systems
/**mqmtop**/qmgrs/QUEUEMANAGERNAME/@ipcc

For Digital OpenVMS systems
mqs_root:[mqm.qmgrs.QM.$IPCC]

The format of this command is:

For OS/2, DOS, Windows 3.1, Windows 95, and Windows NT
SET MQCHLTAB=filename

For UNIX systems
export MQCHLTAB=filename

| For VM/ESA
| GLOBALV SELECT CENV SETLP MQCHLTAB filename

For example:

```
SET MQCHLTAB=ccdf1.tab
```

Note: If you change this environment variable on an MQSeries server, MQSeries will not be able to find any client channel definition table you may have defined before. You must then move your old client channel definition table to the new location.

MQDATA (DOS, Windows 3.1, and Windows NT only)

This holds the path to the directory containing the trace, error and **qm.ini** files. (The **qm.ini** file is needed for setting up NetBIOS.) The default is to root directory of the C drive.

The format of this command is:

SET MQDATA=pathname

The trace and error files are:

AMQERR01.FDC	For First Failure Data Capture messages.
AMQERR01.LOG	For error messages.

An error message will always be added to the end of the log, so the files must be deleted periodically to avoid the files getting too large. At the time a record needs to be added to one of these files, if the file does not exist, it will be created.

These files are written in binary format. Use the **RUNMQFMT** command supplied with MQSeries to reformat these files into a readable form.

MQNAME

This specifies the local NetBIOS name that the MQSeries processes can use. See "Defining a NetBIOS connection" on page 91 for a full description and for the rules of precedence on the client and the server.

The format of this command is:

For OS/2, DOS, Windows 3.1, Windows 95, and Windows NT
SET MQNAME=Your_env_Name

For example:

```
SET MQNAME=CLIENT1
```

The NetBIOS on some platforms requires a different name, set by MQNAME, for each application if you are running multiple MQSeries applications simultaneously on the MQSeries client.

MQSERVER

This is used to define a minimal channel. It specifies the location of the MQSeries server and the communication method to be used. Note that *ConnectionName* must be a fully qualified network name. When the MQSERVER environment variable is used to define a client channel a MAXMSGL of 4 MB is used, so larger messages cannot flow across this channel. For larger messages a CLNTCONN channel must be defined using DEFINE CHANNEL, on the server, with MAXMSGL set to a larger figure.

The format of this command is:

For OS/2, DOS, Windows 3.1, Windows 95, and Windows NT
SET MQSERVER=ChannelName/TransportType/ConnectionName

For OS/2 using LU 6.2
SET MQSERVER=ChannelName/LU62/ConnectionName/ModeName/TpName

For UNIX systems
export MQSERVER=ChannelName/TransportType/ConnectionName

For Digital OpenVMS systems using DECnet
define mqserver "ChannelName/decnet/nodename(object number)"

Or a symbol:
mqserver := "ChannelName/decnet/nodename(object number)"

For VM/ESA systems using TCP/IP

```
GLOBALV SELECT CENV SETLP MQSERVER ChannelName/TCP/ConnectionName
```

For VM/ESA systems using LU 6.2

```
GLOBALV SELECT CENV SETLP MQSERVER ChannelName/LU62/TpName/ModeName
```

TCP/IP default port

By default, for TCP/IP, MQSeries assumes that the channel will be connected to port 1414. You can change this by:

- Adding the port number in brackets as the last part of the ConnectionName:

  For OS/2, DOS, Windows 3.1, Windows 95, and Windows NT
  SET MQSERVER=ChannelName/TransportType/ConnectionName(PortNumber)

  For UNIX systems
  export MQSERVER=ChannelName/TransportType/ConnectionName(PortNumber)

- Changing the `qm.ini` file by adding the port number to the protocol name, for example:

  ```
  TCP:
  port=2001
  ```

- Adding MQSeries to the services file as described in "Defining a TCP/IP connection" on page 77.

SPX default socket

By default, for SPX, MQSeries assumes that the channel will be connected to socket 5E86. You can change this by:

- Adding the socket number in brackets as the last part of the ConnectionName:

  For OS/2, and Windows NT
  SET MQSERVER=ChannelName/TransportType/ConnectionName(SocketNumber)

  For SPX connections, specify the ConnectionName and socket in the form `network.node(socket)`. If the MQSeries client and server are on the same network, the network need not be specified. If you are using the default socket, the socket need not be specified.

- Changing the `qm.ini` file by adding the port number to the protocol name, for example:

  ```
  SPX:
  socket=5E87
  ```

Examples of using MQSERVER

Examples on OS/2:

```
SET MQSERVER=CHAN1/TCP/9.20.4.56(2001)
SET MQSERVER=CHAN1/NETBIOS/BOX643
SET MQSERVER=CHAN1/SPX/000001.08005A7161E5(5E88)
```

Examples on a UNIX system:

```
export MQSERVER=CHAN1/TCP/9.20.4.56(2002)
export MQSERVER=CHAN1/LU62/BOX99/MODENAME1/TPNAME1
```

Examples on Digital OpenVMS:

```
define mqserver "chan1 [DECNET] node(task)"
mqserver="chan1[TCP] 9.20.4.2(2001)"
```

All MQCONN or MQCONNX requests then attempt to use the channel you have defined.

Note: The MQSERVER environment variable takes priority over any client channel definition pointed to by MQCHLLIB and MQCHLTAB, irrespective of any queue manager name specified in an MQCONN or MQCONNX call.

MQTRACE (DOS, Windows 3.1, VM/ESA)

This sets tracing on and off, as required. The default is for tracing to be turned off.

The format of this command is:

For DOS and Windows 3.1
SET MQTRACE=filename,options

| For VM/ESA
| GLOBALV SELECT CENV SETLP MQTRACE filename/options

For example, to direct the communication flow trace entries to *MQ.TRC* file and overwrite the previous trace file each time the program runs:

```
SET MQTRACE=MQ.TRC,cw
```

MQSWORKPATH (OS/2 only)

This specifies the path to the mqs.ini file and is used internally by MQSeries.

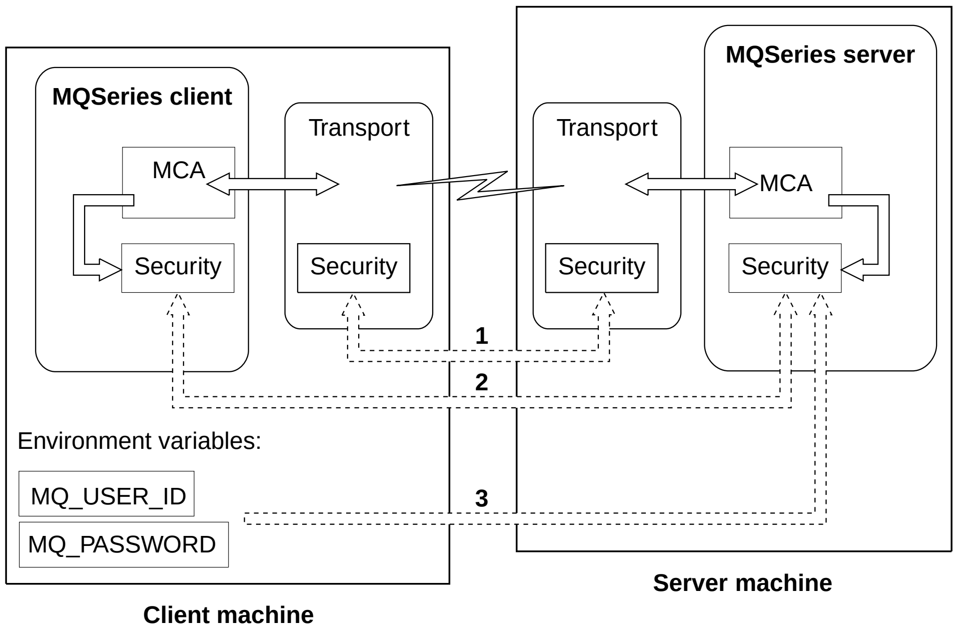
Chapter 9. Setting up MQSeries client security

You must consider MQSeries client security, so that the client applications do not have unrestricted access to resources on the server.

There are two aspects to security between a client application and its queue manager server: authentication and access control.

Authentication

There are three levels of security to consider, as shown in the following diagram. MCA is a Message Channel Agent.



1. Transport level

   This is the same as for two MQSeries queue managers (server to server) and is described in the book *MQSeries Intercommunication*.

2. Channel security exits

   The channel security exits for client to server communication can work in the same way as for server to server communication. A protocol independent pair of exits provide mutual authentication of both the client and the server. A full description is given in the book *MQSeries Intercommunication*.

   DCE security exits are supplied with MQSeries Version 5 products for clients on AIX, HP-UX, OS/2, Sun Solaris, Windows 95, and Windows NT. For details see the book *MQSeries Intercommunication*.

   | You cannot use the supplied DCE security exit from an MQSeries client on
   | Windows 95 connected to an MQSeries for HP-UX server or an MQSeries for
   | Sun Solaris server.

   If no security exits are provided, see “Access control” on page 118 for details.

3. User ID and password passed to a channel security exit

   In client to server communication, the channel security exits do not have to operate as a pair. The exit on the MQSeries client side may be omitted. In this case a user ID and password may be specified via environment variables. These are passed to the channel security exit on the server for authentication. In this case the authentication takes place on the server only and it is not mutual.

   If no security exits are provided, see "Access control" for details.

   Note that the channel security exits are not available for an MQSeries client on a DOS platform, so the environment variables are the only option for MQSeries level authentication.

   Setting these environment variables on an MQSeries client is explained below.

User ID and password

If a security exit is not defined on an MQSeries client, the values of two environment variables MQ_USER_ID and MQ_PASSWORD will be transmitted to the server and will be available to the server security exit in the Channel definition when it is invoked. These values may be used to verify the identity of the MQSeries client.

Set these variables, in the environment in which the MQSeries client is going to run. Note that `MYUSERID` and `MYPASSWORD` must be in uppercase if the MQSeries client is going to communicate with an MQSeries server on AS/400.

On OS/2, DOS, Windows 3.1, Windows 95, and Windows NT:

```
SET MQ_USER_ID=<MYUSERID>
SET MQ_PASSWORD=<MYPASSWORD>
```

On UNIX systems:

```
export MQ_USER_ID=<MYUSERID>
export MQ_PASSWORD=<MYPASSWORD>
```

Access control

Access control in MQSeries is based upon the user identifier associated with the process making MQI calls. For MQSeries clients, the process that issues the MQI calls is the server Message Channel Agent. The user identifier used by the server MCA is that contained in the `MCAUserIdentifier` field of the MQCD. The contents of `MCAUserIdentifier` are determined by the following:

- Any values set by security exits
- MQ_USER_ID environment variable
- MCAUSER (in server-connection channel definition)

Depending upon the combination of settings of the above, `MCAUserIdentifier` is set to the appropriate value. If security exits are provided, `MCAUserIdentifier` may be set by the exit. Otherwise `MCAUserIdentifier` is determined as as shown in the following table:

MQ client ID MQ_USER_ID	**Server channel MCAUSER**	**Value Used**	**Notes**
Not Set or Set	Set	MCAUSER	1
Set	Blanks	MQ_USER_ID	1
Not Set	Blanks	**For MVS/ESA**: The value used is the user ID assigned to the channel initiator started task by the MVS/ESA started procedures table. **For AS/400**: Default User ID QMQM **TCP/IP (non-MVS/ESA)**: User ID from inetd.conf entry **SNA (non-MVS/ESA)**: User ID from SNA Server entry	2,3,4
Not Set or Set	Not Set	**TCP/IP**: User ID from inetd.conf entry **SNA**: User ID from SNA Server entry	2,3,4

Notes:

1. For Windows NT and UNIX servers, the MCAUSER from the channel definition is changed to lowercase before being used. So MCA user identifiers with one or more uppercase letters will **not work** if placed in the MCAUSER field of the channel definition. They will work however if they are put in the client environment variable MQ_USER_ID and MACUSER is blank.
2. For OS/2, no user ID is available from either Communications Manager/2 or NetBIOS.
3. For TCP/IP on Windows NT the value used is the user ID of the person who started the listener.
4. If no user ID is available from the SNA server entry for this transaction program, MCAUser is set to the user ID associated with the incoming attach request, if there is one. (This does not apply for an MCA started as a thread of the MQSeries Listener, that adopts the user ID of the parent process.)
5. For MVS/ESA the channel user ID takes the value of `MCAUserIdentifier` as determined above. See the *MQSeries for MVS/ESA System Management Guide* for more information.

Part 3. Application programming

Chapter 10. Using the message queue interface (MQI)

When you write your MQSeries application, you need to be aware of the differences between running it in an MQSeries client environment and running it in the full MQSeries queue manager environment.

This chapter explains the things to consider.

Limiting the size of a message

The maximum message length (MaxMsgLength) attribute of a queue manager is the maximum length of a message that can be handled by that queue manager. The default maximum message length supported depends on the platform you are using. Details are given in the *MQSeries Application Programming Guide*.

You can find out the value of MaxMsgLength for a queue manager by using the MQINQ call.

If the MaxMsgLength attribute is changed, no check is made that there are not already queues, and even messages, with a length greater than the new value. After a change to this attribute, applications and channels should be restarted in order to ensure that the change has taken effect. It will then not be possible for any new messages to be generated that exceed either the queue manager's MaxMsgLength or the queue's MaxMsgLength (unless queue manager segmentation is allowed).

The maximum message length in a channel definition limits the size of a message that you can transmit along a client connection. If an MQSeries application tries to use the MQPUT call or the MQGET call with a message larger than this, an error code is returned to the application.

Choosing client or server coded character set identifier (CCSID)

The data passed across the MQI from the application to the client stub should be in the local CCSID (coded character set identifier), encoded for the MQSeries client. If the connected queue manager requires the data to be converted, this will be done by the client support code.

The client code will assume that the character data crossing the MQI in the client is in the CCSID configured for that machine. For example, if this CCSID is an unsupported CCSID or is not the required CCSID, it can be overridden with the MQCCSID environment variable, for example:

```
SET MQCCSID=850
```

Or, on UNIX systems: `export MQCCSID=850`

Set this in the profile and all MQI data will be assumed to be in code page 850.

Note: This does not apply to application data in the message.

Controlling application in a Windows 3.1 environment

A Windows 3.1 MQSeries client runs within a full Windows environment, not under a DOS prompt.

Normally, when you issue a request from a non-MQSeries application, control is not returned to that application until the request is fulfilled. This is because the Windows 3.1 environment is a cooperative multi-tasking system.

However, the MQSeries client code overrides the locking of the machine and the application to enable you to start up more applications, or work on something else until the MQI call has been answered. Should the application attempt to issue a further MQI call before the previous one has been answered, the application will get a return code indicating that there is still a call in progress, and the second call will fail.

Designing applications

When designing an application, consider what controls you need to impose during an MQI call to ensure that the MQSeries application processing is not disrupted.

Windows 3.1 environment

To cooperate fully in the Windows 3.1 multi-tasking environment, an MQI call results in the client code executing a GetMessage loop on behalf of the application. If an application has accelerator keys defined, these will not function until the MQI call returns and control is returned to the GetMessage loop of the application.

Note: There is only one way that an ongoing MQI call can be cancelled - by the application receiving a WM_QUIT message.

Using MQINQ

Some values queried using MQINQ will be modified by the client code.

CCSID
: is set to the client CCSID, not that of the queue manager.

MaxMsgLength
: is reduced if it is restricted by the channel definition. This will be the lower of:

- The value defined in the queue definition, or
- The value defined in the channel definition

Using syncpoint coordination

Within MQSeries, one of the roles of the queue manager is syncpoint control within an application. If an application runs on an MQSeries client, then it can issue MQCMIT and MQBACK, but the scope of the syncpoint control is limited to the MQI resources.

Applications running in the full queue manager environment, on the server, can coordinate multiple resources (for example databases) via a transaction monitor. On the server you can use the Transaction Monitor supplied with the Version 5 MQSeries products, or another transaction monitor such as CICS. You cannot use a transaction monitor with a client application. The MQSeries verb MQBEGIN is not valid in a client environment.

MQSeries for Tandem NSK server

When an MQSeries client connects to a queue manager on MQSeries for Tandem NSK V2.2:

- Any MQGET, MQPUT, or MQPUT1 with an MQ*_SYNCPOINT option initiates a Tandem transaction, if one has not already been associated with the connection handle.
- Any MQGET, MQPUT, or MQPUT1 with neither an MQ*_SYNCPOINT nor an MQ*_NO_SYNCPOINT option initiates a Tandem transaction, if one has not already been associated with the connection handle.
- The MQCMIT call commits a Tandem transaction, if one is associated with the connection handle. The MQBACK call cancels the Tandem transaction, if one is associated with the connection handle.

In all cases, if the Tandem BEGINTRANSACTION fails, a *CompCode* of MQCC_FAILED, and a *Reason* of MQRC_SYNCPOINT_NOT_AVAILABLE are returned to the caller.

Using MQCONNX

MQCONNX can be used from a client but these options are ignored:

MQCNO_STANDARD_BINDING
MQCNO_FASTPATH_BINDING

MQCONN and MQCONNX on a client are equivalent calls. The actual call issued at the server depends on the server level and the listener configuration.

Chapter 11. Building applications for MQSeries clients

If an application is to run in a client environment you can write it in C, COBOL, or C++ (C only, for DOS, Windows 3.1, and AT&T GIS UNIX). You can write the application in PL/I for AIX, OS/2, and Windows NT. You can use Java applets if you have installed the MQSeries client for Java on AIX, HP-UX, Sun Solaris, OS/2, and Windows NT. You must link your application to the relevant client library file.

This chapter lists points to consider when running an application in an MQSeries client environment, and describes how to link your application code with the MQSeries client code.

Note for C++ users: If you are using an MQSeries client supplied with an MQSeries Version 5 product you can write your applications to run on the client in C++. Programs that use the MQSeries C++ classes can be used successfully with MQSeries Version 5 or AS/400 V4R2 servers only. To see how to link your C++ applications and for full details of all aspects of using C++ see the book *MQSeries Using C++*.

Note for Java users: If you have installed the MQSeries client for Java, you can write Java applets that communicate with MQSeries. For full details, see the MQSeries Internet/ Java documentation. This is supplied as a client component that you can install, and is in HTML format. The client READ.ME file on the client CD-ROM tells you where to point your browser to read this documentation.

Running applications in the MQSeries client environment

You can run an MQSeries application both in a full MQSeries environment and in an MQSeries client environment without changing your code, providing:

- It does not need to connect to more than one queue manager concurrently
- The queue manager name is not prefixed with an asterisk (*) on an MQCONN or MQCONNX call

Note: The libraries you use at link-edit time determine the environment in which your application must run.

When working in the MQSeries client environment, remember:

- Each application running in the MQSeries client environment has its own connections to servers. It will have one connection to each server it requires, a connection being established with each MQCONN or MQCONNX call the application issues.
- An application sends and gets messages synchronously.
- All data conversion is done by the server, but see also “MQCCSID” on page 112.
- Triggering, see “Triggering in the client environment” on page 128.

Triggering in the client environment

Triggering is explained in detail in the *MQSeries Application Programming Guide*.

Messages sent by MQSeries applications running on MQSeries clients contribute to triggering in exactly the same way as any other messages, and they can be used to trigger programs on the server. The trigger monitor and the application to be started must be on the same system.

Process definition

You must define the PROCESS definition on the server, as this is associated with the queue that has triggering set on.

The process object defines what is to be triggered. If the client and server are not running on the same platform, any processes started by the trigger monitor must define *ApplType*, otherwise the server takes its default definitions (that is, the type of application that is normally associated with the server machine) and causes a failure.

For example, if the trigger monitor is running on a Windows NT client and wants to send a request to an OS/2 server, MQAT_WINDOWS_NT must be defined otherwise OS/2 uses its default definitions (that is, MQAT_OS2) and the process fails.

For a list of application types, see the *MQSeries Application Programming Reference* manual.

Client on Windows 95.

The Windows 95 client runs in 32-bit mode. It is also possible to run the client for Windows 3.1 in 16-bit mode on a Windows 95 platform. If a trigger monitor is running on a Windows 95 client you must make sure that you define the correct *ApplType*:

MQAT_WINDOWS
Windows 3.1 client or 16-bit Windows application.

MQAT_WINDOWS_NT
Windows NT client or 32-bit Windows application.

Trigger monitor

The trigger monitor provided by non-MVS/ESA MQSeries products runs in the Digital OpenVMS, OS/2, Windows 3.1, Windows 95, Windows NT, and UNIX systems MQSeries client environments. To run it, issue the command:

```
runmqtmc [-m QMgrName] [-q InitQ]
```

The default initiation queue is SYSTEM.DEFAULT.INITIATION.QUEUE on the default queue manager. This is where the trigger monitor looks for trigger messages. It then calls programs for the appropriate trigger messages. This trigger monitor supports the default application type and is the same as `runmqtrm` except that it links the client libraries.

The command string, built by the trigger monitor, is as follows:

1. The `applicid` from the relevant PROCESS definition. This is the name of the program to run, as it would be entered on the command line.
2. The `MQTMC2` structure, enclosed in quotes, as got from the initiation queue. A command string is invoked which has this string, exactly as provided, in 'quotes', in order that the system command will accept it as one parameter.
3. The `envrdata` from the relevant PROCESS definition.

The trigger monitor will not look to see if there is another message on the initiation queue until the completion of the application it has just started. If the application has a lot of processing to do, this may mean that the trigger monitor cannot keep up with the number of trigger messages arriving. You have two options:

- Have more trigger monitors running
- Run the started applications in the background

If you choose to have more trigger monitors running, you can control the maximum number of applications that can run at any one time. If you choose to run applications in the background, MQSeries imposes no restriction on the number of applications that can run.

To run the started application in the background on an OS/2 system, within the `applicid` field you must prefix the name of your application with a `start` command; for example, `start amqsinq /B`.

To run the started application in the background on a UNIX system, you must put an & (ampersand) at the end of the `envrdata` of the PROCESS definition.

CICS applications (non-MVS/ESA)

A non-MVS/ESA CICS application program that issues an MQCONN or MQCONNX call must be defined to CEDA as RESIDENT. To make the resident code as small as possible, it may be worth linking to a separate program to issue the MQCONN or MQCONNX call.

If the MQSERVER environment variable is used to define the client connection, it must be specified in the CICSENV.CMD file.

MQSeries applications can be run in an MQSeries server environment or on an MQSeries client without changing code. However, in an MQSeries server environment, CICS can act as syncpoint coordinator, and you use EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK rather than MQCMIT and MQBACK. If a CICS application is simply relinked as a client, syncpoint support is lost. MQCMIT and MQBACK must be used for the application running on an MQSeries client.

Channel exits

The channel exits available to the MQSeries client environment, in OS/2, UNIX systems, Windows 3.1, Windows 95, and Windows NT environments are:

- Send exit
- Receive exit
- Security exit

| The channel exits are not available on DOS systems.

These exits are available at both the client and server ends of the channel.

Remember, exits are not available to your application if you are using the MQSERVER environment variable. Exits are explained in the book *MQSeries Intercommunication*.

The send and receive exit work together. There are several possible ways in which you may choose to use them:

- Splitting and reassembling a message
- Compressing and decompressing data in a message
- Encrypting and decrypting user data
- Journaling each message sent and received

You can use the security exit to ensure that the MQSeries client and server machines are correctly identified, as well as to control access to each machine.

Path to exits

This applies to MQSeries clients on AIX, HP-UX, OS/2, Sun Solaris, Windows NT, and Windows 95 systems.

An `mqs.ini` file is added to your system during installation of the MQSeries client. A default path for location of the channel exits on the client is defined in this file, using the stanza:

```
ClientExitPath:
  ExitsDefaultPath=<defaultprefix>/exits
```

Where `<defaultprefix>` is the value defined for your system in the DefaultPrefix stanza of the `mqs.ini` file.

When a channel is initialized, after an MQCONN or MQCONNX call, the `mqs.ini` file is searched. The ClientExitPath stanza is read and any channel exits that are specified in the channel definition are loaded.

Linking C applications with the MQSeries client code

Having written your MQSeries application that you want to run on the MQSeries client, you must link it to a queue manager. You can do this in two ways:

- Directly, in which case the queue manager must be on the same machine as your application
- To a client library file, which gives you access to queue managers on the same or on a different machine

MQSeries provides a client library file for each environment:

AIX

libmqic.a library for threaded applications, or

libmqic_r.a library for non-threaded applications.

AT&T GIS UNIX

libmqic.so and libmqmcs.so

If you want to use the programs on a machine that has only the MQSeries client for AT&T GIS UNIX installed, you must recompile the programs to link them with the client library:

```
$ /bin/cc -o <prog> <prog>.c -lmqic -lmqmcs -lmqzse -lnet \
  -lnsl -lsocket -ldl
```

Digital OpenVMS
MQIC.EXE in SYS$SHARE

DOS
MQIC.LIB (see also “For DOS only” on page 132)

HP-UX
libmqic.sl

OS/2
MQIC.LIB

SINIX and DC/OSx
libmqic.so and libmqmcs.so

If you want to use the programs on a machine that has only the MQSeries client for SINIX and DC/OSx installed, you must recompile the programs to link them with the client library:

```
$cc -o <prog> <prog>.c -lmqic -lmqmcs -lmqmzse -lnsl \
  -lsocket -ldl -lmproc -lext
```

For DC/OSx append `-liconv` to the above command line.

Sun Solaris
libmqic.so and libmqmcs.so

If you want to use the programs on a machine that has only the MQSeries client for Sun Solaris installed, you must recompile the programs to link them with the client library:

```
$ /opt/SUNWspro/bin/cc -o <prog> <prog> c -mt -lmqic \
-lmqmcs -lsocket -lc -lnsl -ldl
```

SunOS
libmqic.so and libmqmcs.so

Windows 3.1
MQIC.LIB

| **Windows NT**
| MQIC32.LIB

Windows 95
MQIC32.LIB

| Running 16 and 32-bit Windows clients.

| Previous versions of the MQSeries clients for Windows NT and Windows 95
| included a version of MQIC.DLL that was 32-bit. When client code that was
| compiled as 16-bit is run using this DLL, it fails due to a name clash caused by the
| file MQIC.DLL. This has been rectified by replacing MQIC.DLL with MQIC32.DLL.

| The file MQIC.DLL is no longer included in the 32-bit Windows client. If you have
| code linked with the MQIC.DLL you need to relink. If this is not possible, you can
| copy the MQIC32.DLL file to MQIC.DLL. Note that doing this will prevent you from
| running mixed 16 and 32-bit environments.

For DOS only

Your application must also be linked with at least three of the following libraries, one for each protocol, indicating whether you do or do not require it.

MQICN	NetBIOS required
MQICDN	NetBIOS not required
MQICS	SPX required
MQICDS	SPX not required
MQICT	TCP/IP required
MQICDT	TCP/IP not required
SOCKETL	Link to this from the DOS TCP/IP product (if using TCP/IP)

When compiling programs in these environments there are many options available. For example, using Microsoft C7:

```
/Alfw /Gw /Zpl /J
```

with a stack size greater than 8 KB, preferably 16 KB.

Linking C++ applications with the MQSeries client code

If you are using an MQSeries client supplied with an MQSeries Version 5 product
| you can write your applications to run on the client in C++. Programs that use the
| MQSeries C++ classes can be used successfully with MQSeries Version 5 or
| AS/400 V4R2 servers only. For how to link your C++ applications and for full details of all aspects of using C++ see the book *MQSeries Using C++*.

Linking COBOL applications with the MQSeries client code

AIX
Link your COBOL application with the libmqicb.a library.

AT&T GIS UNIX
There is no COBOL support on AT&T GIS UNIX.

Digital OpenVMS
Link your COBOL application with the MQICB.EXE library in SYS$SHARE.

HP-UX
Link your COBOL application with the libmqicb.sl library.

If you are not using LU 6.2, consider linking to libsnastubs.a (in /opt/lib for HP-UX) to fully resolve function names. The need to link to this library varies with how you are using the -B flag during the linking stage. For more information see the *MQSeries Application Programming Guide*.

OS/2
If you have an OS/2 COBOL application that you want to run in the client environment, link your application code with the MQICCB16 library for 16-bit COBOL, or the MQICCBB library for 32-bit COBOL.

As with any MQSeries application, you must compile it with the LITLINK directive. The COBLIB library must appear before the DOSCALLS library in the library list, and you need a stack size greater than 8 KB.

Additionally, your application needs a runtime stack size of at least 16 KB. More may be required depending on your application. One way to set this is to use the COBSW environment variable. For example:

```
set COBSW=/S16384
```

This stack size is sufficient to run the sample COBOL applications as clients.

SINIX and DC/OSx

If you have a COBOL application that you want to run in the client environment, you must recompile the programs to link them with the client library, libmqmcb.so:

```
cob -xU <prog>.cbl -lmqmcb -lmqm -mqmcs -mqmzse -lmproc
```

For DC/OSx append `liconv` to the above command line.

Note: `-lmqmcb` **must** come before `-lmqm` on the command line.

SunOS

Link your COBOL application with the libmqicb library.

If you want to use LU 6.2 ***do not*** link your application with the Sunlink 7.0 libp2p.a library. The required modules from libp2p.a are linked into the MQSeries client library. ***Do not*** use the libsnastubs.a or the Sunlink 7.0 libp2p.a libraries. All modules required for SNA LU 6.2 connectivity are in the MQSeries client library libmqicb.a.

Sun Solaris

Link your COBOL application with the libmqicb.so library.

Windows NT and Windows 95

If you have a Windows NT, or a Windows 95 COBOL application that you want to run in the client environment, link your application code with the MQICCBB library for 32-bit COBOL. MQSeries for Windows NT and the MQSeries Windows 95 client do not support 16-bit COBOL.

Linking PL/I applications with the MQSeries client code

See the *MQSeries Application Programming Guide* for further details.

AIX

Link your PL/I application with:

- libmqic.a library for threaded applications, or
- libmqic_r.a library for non-threaded applications.

OS/2

Link your PL/I application with the MQIC.LIB library.

Windows NT

Link your PL/I application with the MQIC32.LIB library.

Chapter 12. Running applications on MQSeries clients

This chapter explains the various ways in which an application running in an MQSeries client environment can connect to a queue manager. It covers the relationship of the MQSERVER environment variable provided by MQSeries, and the client channel definition table created by MQSeries.

When an application running in an MQSeries client environment issues an MQCONN or MQCONNX call, the client code identifies how it is to make the connection:

1. If the MQSERVER environment variable is set, the channel it defines will be used.
2. If the MQCHLLIB and MQCHLTAB environment variables are set, the client channel definition table they point to will be used.
3. Finally, if the environment variables are ***not*** set, the client code searches for a channel definition table whose path and name are established from the `DefaultPrefix` in the `mqs.ini` file. If this fails, the client code will use the paths:

OS/2	`MQSWORKPATH\amqclchl.tab`
	Or, if MQSWORKPATH is not set:
	`bootdrive:\mqm\amqclchl.tab`
UNIX systems	`/var/mqm/AMQCLCHL.TAB`
Windows NT and Windows 95	`bootdrive:\mqm\amqclchl.tab`
	Where `bootdrive` is obtained from the `Software\IBM\MQSeries\CurrentVersion` registry entry under `HKEY_LOCAL_MACHINE`. This value is established when the MQSeries client software is installed. If it is not found, a value of 'C' is used for `bootdrive`.
Digital OpenVMS	mqs_root:[mqm]amqclchl.tab

Notes:

1. If the client code fails to find any of these, the MQCONN or MQCONNX call will fail.
2. The channel name, established from either the first segment of the MQSERVER variable or from the client channel definition table, must match the SVRCONN channel name defined on the server for the MQCONN or MQCONNX call to succeed.
3. If you receive a MQRC_Q_MGR_NOT_AVAILABLE return code from your application with an error message in the error log file of AMQ9517 - File damaged, see "Migrating from MQSeries for OS/2 V2.0 and MQSeries for AIX V2.1 or V2.2" on page 108.

Using MQSERVER

If you use the MQSERVER environment variable to define the channel between your MQSeries client machine and a server machine, this is the only channel available to your application, and no reference is made to the client channel definition table. In this situation, the 'listener' program that you have running on the server machine determines the queue manager to which your application will connect. It will be the same queue manager as the listener program is connected to.

If the MQCONN or MQCONNX request specifies a queue manager other than the one the listener is connected to, or if TransportType is not recognized, the MQCONN or MQCONNX request fails with return code MQRC_Q_MGR_NAME_ERROR.

Using DEFINE CHANNEL

If you use the MQSC **DEFINE CHANNEL** command, the details you provide are placed in the client channel definition table. It is this file that the client code accesses, in channel name sequence, to determine the channel an application will use.

The contents of the `Name` parameter of the MQCONN or MQCONNX call determines which server the client connects to.

Role of the client channel definition table

The client channel definition table is created when you define the first of the connections between an MQSeries client and a server. See "Connecting the MQSeries client and server - channel definitions" on page 100 for more information on what you have to define and how you do it.

Note: The same file may be used by more than one MQSeries client. You change the name and location of this file using the MQCHLLIB and MQCHLTAB MQSeries environment variables. See Chapter 8, "Using MQSeries environment variables" on page 111 for details of these and all the other MQSeries environment variables.

Multiple queue managers

You may choose to define connections to more than one server machine because:

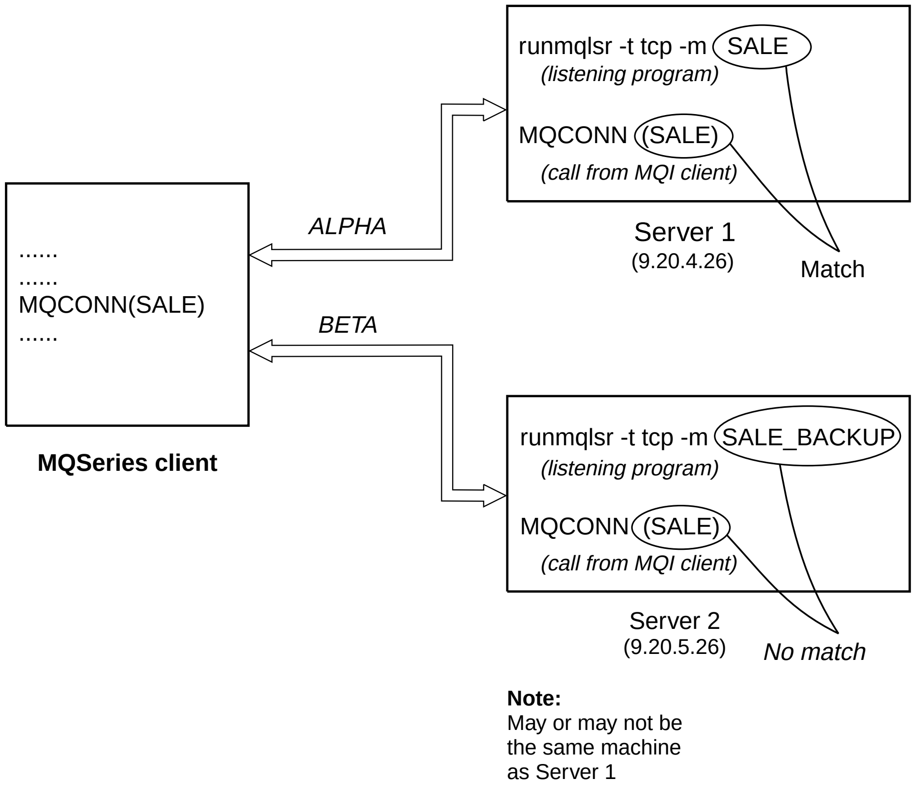
- You need a backup system.
- You want to be able to move your queue managers without changing any application code.
- You need to access multiple queue managers, and this requires the least resource.

Note: Define your client-connection and server-connection channels on one queue manager only, including those channels that connect to a second or third queue manager. Do ***not*** define them on two queue managers and then try to merge the two client channel definition tables; this cannot be done. Only one client channel definition table can be accessed by the client.

Examples of MQCONN calls

Note: In these examples, the MQCONNX call could be used instead of the MQCONN call.

In each of the following examples, the network is the same; there is a connection defined to two servers from the same MQSeries client. There are two queue managers running on the server machines, one named `SALE` and the other named `SALE_BACKUP`.



The definitions for the channels in these examples are:

SALE Definitions:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to MQSeries client')

DEFINE CHANNEL(APLHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) DESCR('MQSeries client connection to server 1') +
QMNAME(SALE)

DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.5.26) DESCR('MQSeries client connection to server 2') +
QMNAME(SALE)
```

SALE_BACKUP Definition:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to MQSeries client')
```

The client channel definitions may be summarized as follows:

Name	CHLTYPE	TRPTYPE	CONNAME	QMNAME
ALPHA	CLNTCONN	TCP	9.20.4.26	SALE
BETA	CLTCONN	TCP	9.20.5.26	SALE

What the Examples demonstrate

Suppose the communication link to Server 1 is temporarily broken. The use of multiple queue managers as a backup system is demonstrated.

Each example covers a different MQCONN call and gives an explanation of what happens in the specific example presented, by applying the following rules:

1. MQSeries searches the client channel definition table, in ***channel name order***, looking in the queue manager name (QMNAME) field for an entry corresponding to the one given in the MQCONN call.
2. If a match is found the transmission protocol and the associated connection name are extracted.
3. An attempt is made to start the channel to the machine, identified by the connection name (CONNAME). If this is successful, the application will continue. It requires:
   - A listener to be running on the server
   - The listener to be connected to the same queue manager as the one the client wishes to connect to (if specified)
4. If the attempt to start the channel fails and there is more than one entry in the client channel definition table (in this example there are two entries), the file is searched for a further match. If a match is found, processing continues at step 1.
5. If no match is found, or there are no more entries in the client channel definition table and the channel has failed to start, the application is unable to connect. An appropriate reason code and completion code are returned in the MQCONN call. The application can take action based on the reason and completion codes returned.

Example 1. Queue manager name prefixed with an asterisk (*)

In this example the application is not concerned about which queue manager it connects to. The application issues:

```
MQCONN (*SALE)
```

Following the rules, this is what will happen in this instance:

1. The client channel definition table is scanned (in channel name order) for the queue manager name `SALE`, matching with the application MQCONN call.
2. The first channel definition found to match is `ALPHA`.
3. An attempt to start the channel is made – this is NOT successful because the communication link is broken.
4. The client channel definition table is again scanned for the the queue manager name `SALE` and the channel name `BETA` is found.
5. An attempt to start the channel is made – this is successful.
6. A check to see that a listener is running shows that there is one running. It is not connected to the `SALE` queue manager, but because the MQI call parameter has an asterisk (*) in front of it, no check is made. The application will be connected to the `SALE_BACKUP` queue manager and will continue processing.

Example 2. Queue manager name specified

The application requires a connection to a specific queue manager, named `SALE`, as seen in the MQI call:

```
MQCONN (SALE)
```

Following the rules, this is what will happen in this instance:

1. The client channel definition table is scanned (in channel name order) for the queue manager name `SALE`, matching with the application MQCONN call.
2. The first channel definition found to match is `ALPHA`.
3. An attempt to start the channel is made – this is NOT successful because the communication link is broken.
4. The client channel definition table is again scanned for the the queue manager name `SALE` and the channel name `BETA` is found.
5. An attempt to start the channel is made – this is successful.
6. A check to see that a listener is running shows that there is one running, but it is not connected to the `SALE` queue manager.
7. There are no further entries in the client channel definition table. The application cannot continue and will receive error number 2059 - Queue Manager unavailable.

Example 3. Queue manager name is blank or an asterisk (*)

In this example the application is not concerned about which queue manager it connects to. This is treated in the same way as the previous example.

Note: If this application were running in an environment other than an MQSeries client, and the name was blank, it would be attempting to connect to the default queue manager. This is ***not*** the case when it is run from a client environment, as there can be more than one default queue manager. The application issues:

```
MQCONN ("")
MQCONN (*)
```

Following the rules, this is what will happen in this instance:

1. The client channel definition table is scanned (in channel name order) for a queue manager name that is blank, matching with the application MQCONN call.
2. The entry for the channel name `ALPHA`. has a queue manager name in the definition of `SALE`. This does ***not*** match the MQCONN call parameter, which requires the queue manager name to be blank.
3. The next entry is for the channel name `BETA`.
4. The `queue manager name` in the definition is `SALE`. Once again, this does ***not*** match the MQCONN call parameter, which requires the queue manager name to be blank.
5. There are no further entries in the client channel definition table. The application cannot continue and will receive error number 2058 - Queue Manager name error.

Chapter 13. Solving problems

This chapter discusses the return codes, error logs, and error messages. It examines some common problems when running applications in the MQSeries client environment. Trace tools by platform are also covered.

An application running in the MQSeries client environment receives MQRC_* reason codes in the same way as MQSeries server applications. However, there are additional reason codes for error conditions associated with MQSeries clients. For example:

- Remote machine not responding
- Communications line error
- Invalid machine address

The most common time for errors to occur is when an application issues an MQCONN or MQCONNX and receives the response MQRC_Q_MQR_NOT_AVAILABLE. Look in the client error log for a message explaining the failure. There may also be errors logged at the server, depending on the nature of the failure. Also, check that the application on the MQSeries client is linked with the correct library file.

MQSeries client fails to make a connection

When the MQSeries client issues an MQCONN or MQCONNX call to a server, socket and port information is exchanged between the MQSeries client and the server. For any exchange of information to take place, there must be a program on the server machine whose role is to ‘listen’ on the communications line for any activity. If there is no program doing this, or there is one but it has problems of its own, the MQCONN or MQCONNX call fails and the relevant reason code is returned to the MQSeries application.

If the connection is successful, MQSeries protocol messages are then exchanged and further checking takes place. It is not until all these checks are successful that the MQCONN or MQCONNX call will succeed.

During the MQSeries protocol checking phase, some aspects are negotiated while others cause the connection to fail.

For full details of the MQRC_* reason codes, see the *MQSeries Application Programming Reference*.

Stopping MQSeries clients

Even though an MQSeries client has stopped, it is still possible for the process at the server to be holding its queues open. The queues will be closed when the communications layer detects that the partner has gone.

Error messages with MQSeries clients

When an error occurs with an MQSeries client system, error messages are put into the error files associated with the server, if possible. If the error cannot be placed there, the MQSeries client code attempts to place the error message in an error log in the root directory of the MQSeries client machine.

Digital OpenVMS, OS/2, Windows 95, Windows NT, and UNIX systems

Error messages for MQSeries clients on Digital OpenVMS OS/2, Windows 95, Windows NT, and UNIX systems are placed in the error logs in the same way they are for the respective MQSeries server systems. Typically these files appear in `/var/mqm/errors` for UNIX systems, in `/mqm/errors` for OS/2, Windows 95, and Windows NT systems, and in the MQS_ROOT:[MQM.ERRORS] directory for Digital OpenVMS systems.

DOS and Windows 3.1 clients

The log file **AMQERR01.LOG** is held on C:\ unless the MQDATA environment variable is used to override the default. See Chapter 8, “Using MQSeries environment variables” on page 111 for details on how to use this and all other MQSeries environment variables.

How to read the error log and FFDCs for DOS and Windows 3.1

| RUNMQFMT reformats the trace, error, and FFDC files. Before running RUNMQFMT you must have access to the error message file, amq9.msg. You can give RUNMQFMT access either by putting the file in the local directory or by adding its location to the DOS APPEND statement.

RUNMQFMT has one optional parameter, the name of the file to be processed. If you do not specify a filename and tracing is on, FORMAT TRACE/ERROR FILE attempts to format the trace file; if tracing is not on, it attempts to format the error file. The output is written to *stdout* to enable you to browse it; alternatively you can redirect the output to a printer. The oldest message is listed first.

To print the output file: `RUNMQFMT filename > printername`

Note: The normal default ‘printer name’ is `LPT1`, which is the port assigned to a printer. Alternatively, you can direct the output to a file, replacing the printer name with the file name when you issue the command.

Your application program should handle any MQI reason codes to allow your program to end in a controlled manner, as there is no MQI error handling within the product.

There are three ways of using RUNMQFMT:

1. Specify the full path and name of the error log.
2. Specify the name of the error log, in which case the default path is used.
3. Enter only the command name, RUNMQFMT. The command assumes that the error log is in the default location, unless this has been changed by the MQDATA environment variable. If tracing is on, the trace is formatted; if tracing is off, the error log is formatted.

MQSeries environment variables

For details of all the environment variables that can be used, see Chapter 8, “Using MQSeries environment variables” on page 111.

MQSeries uses default values for those variables that you have not set. Issue the command from the command line to make a change for your current session only; or, if you want one or more variables to have a particular value dependent on the application that is running, you can add `SET` commands to a command file (.cmd) used by the application. Update your system profile to make a permanent change.

Using trace on DOS and Windows 3.1

Use the MQTRACE environment variable to set tracing on (see “MQTRACE (DOS, Windows 3.1, VM/ESA)” on page 116). Specify the name of the file to which you want all the trace entries to be put. You can further define the use of this file by specifying flags:

c Trace the communications flow.

m Do not query the configuration of the machine your application is running on. Use this option if exceptions occur when normal tracing is switched on.

w Write a new instance of the trace file for each program. If this is not set, the trace entries continue to be added to a single trace file.

Note: If you are using this trace option on two or more applications at the same time, you must specify a different trace filename for each of the applications to guarantee that no trace entries are lost.

Figure 2 on page 144 shows an example.

Example DOS trace data

The following example shows an extract from a trace for a DOS MQSeries client:

```
Trace started on Mon Sep 19 10:48:42 1994
PS/2 Mod 80 or 95 - DOS V20.10 (Rev.2) RAM [BIOS Rev.5]
10:48:42  MQCONN
           rrxOpenChannelDef
           rrxOpenChannelDef RC=0 OK
           rrxGetFirstChannelDef
           rrxGetFirstChannelDef RC=0 OK
           rriInitSess
            rriAddStatusEntry
            rriAddStatusEntry RC=0 OK
            rriInitExits
            rriInitExits RC=0 OK
            ccxNetWorkInit
 ...

 ...
              ccxQueryProcAddr RC=0 OK
             cciLoadLibrary RC=0 OK
            ccxNetWorkInit RC=0 OK
            ccxAllocConv
             cciNetbAllocConv
             cciNetbAllocConv RC=0 OK
            ccxAllocConv RC=0 OK
            ccxAllocMem
            ccxAllocMem RC=0 OK
            ccxSend
             cciNetbSend
 ---------------------------------------------------------------------
I 10:48:53  Outbound 72 bytes.
I  54534820 00000048 02010100 00000000    TSH....H........
I  00000000 22020000 52030000 49442020    ........R...ID..
I  02250000 00000000 FE0F0000 00004000    ................
I  00000000 4F533250 4743312E 53525620    ....OS2PGC1.SRV.
I  20202020 20202020                      ........
 ---------------------------------------------------------------------
             cciNetbSend RC=0 OK
            ccxSend RC=0 OK
 ...
```

Figure 2. Extract from a DOS client trace

The entries in the box represent data sent or received over communications links.

Using trace on OS/2, Windows NT, and Windows 95

MQSeries for Windows NT and MQSeries for OS/2 use the following commands for the MQSeries client trace facility:

strmqtrc to start early tracing
endmqtrc to end tracing

An MQSeries client on Windows 95 uses the following commands for the MQSeries client trace facility:

strmqtrc -t(TraceType) to start tracing
endmqtrc to end tracing

File names for trace files

Trace file names are constructed in the following way:

```
AMQppppp.TRC
```

where ppppp is the process ID (PID) of the process producing the trace.

Notes:

1. The value of the process id can contain fewer or more digits than shown in the example.
2. There will be one trace file for each process running as part of the entity being traced.

How to examine FFSTs

The files are produced already formatted and are in the `\mqm\errors` directory.

These are normally severe, unrecoverable errors and indicate either a configuration problem with the system or an MQSeries internal error.

The files are named `AMQnnnnn.mm.FDC`, where:

`nnnnn` is the process id reporting the error
`mm` is a sequence number, normally 0

When a process creates an FFST it also sends a record to syslog. The record contains the name of the FFST file to assist in automatic problem tracking.

The syslog entry is made at the “user.error” level.

The MQSeries trace utility is explained in detail in the book *MQSeries System Administration*.

Using trace on AIX and AT&T GIS UNIX

MQSeries for AIX and MQSeries for AT&T GIS UNIX use the standard UNIX system trace. Tracing is a two step process:

1. Gather the data.
2. Format the results.

MQSeries uses two trace hook identifiers:

X'30D'	This event is recorded by MQSeries on entry to or exit from a subroutine.
X'30E'	This event is recorded by MQSeries to trace data such as that being sent or received across a communications network.

Trace provides detailed execution tracing to help you to analyze problems. IBM service support personnel may ask for a problem to be recreated with trace enabled. The files produced by trace can be ***very*** large, so it is important to qualify a trace, where possible. For example, you can optionally qualify a trace by time and by component.

Activate a trace using the standard AIX trace command. For example:

```
trace -a -j30D,30E -o tracefile_name
```

The binary trace is run asynchronously and is stored in the specified file. When you get the shell prompt, enter the command to start the operation to be traced, prefixed by **!**. For example:

```
!trace -a -j30D,30E -o /u/userid/trace/first.dat
```

When you want to stop, type **trcstop** to end the trace. Then format the trace file with the command:

```
trcrpt -t /usr/lpp/mqm/lib/amqtrc.fmt first.dat > report_name
```

`report_name` is the name of the file where you want to put the formatted trace output.

Note: ***All*** MQSeries activity on the machine is traced while the trace is active. If you have only an MQSeries client installed on your machine, only the activity of the MQSeries client is traced.

The MQSeries trace utility is explained in detail in the *MQSeries System Administration* book (for AIX) and the *MQSeries for AT&T GIS UNIX System Management Guide*.

Using trace on Digital OpenVMS, HP-UX, SINIX, DC/OSx, SunOS, and Sun Solaris

| MQSeries for Digital OpenVMS HP-UX, SINIX and DC/OSx, SunOS, and Sun Solaris use the following commands for the MQSeries client trace facility:

| **strmqtrc -e** to start early tracing
| **endmqtrc -e** to end early tracing
| **dspmqtrc** <**filename**> to display a formatted trace file

| For more information about the trace commands, see the *MQSeries System*
| *Administration* book for Version 5 products, or the *System Management Guide* for
| your platform for non-Version 5 products.

The trace facility uses a number of files, which are:

- One file for each entity being traced, in which trace information is recorded

- One additional file on each machine, to provide a reference for the shared memory used to start and end tracing
- One file to identify the semaphore used when updating the shared memory

Files associated with trace are created in a fixed location in the file tree, which is `/var/mqm/trace`.

| On Digital OpenVMS systems the files are in the MQS_ROOT:[MQM.ERRORS]
| directory.

All queue managers tracing, all early tracing and all @SYSTEM tracing takes place to files in this directory.

Note: You can handle large trace files by mounting a temporary filesystem over this directory.

File names for trace files

Trace file names are constructed in the following way:

```
AMQppppp.TRC
```

where ppppp is the process ID (PID) of the process producing the trace.

Notes:

1. The value of the process id can contain fewer or more digits than shown in the example.
2. There will be one trace file for each process running as part of the entity being traced.

How to examine FFSTs

Information that, on the OS/2 and AIX platforms, is normally recorded in FFST logs is, on HP-UX, AT&T GIS UNIX, SINIX, DC/OSx, SunOS, and Sun Solaris recorded in a file in the `/var/mqm/errors` directory.

These are normally severe, unrecoverable errors and indicate either a configuration problem with the system or an MQSeries internal error.

The files are named `AMQnnnnn.mm.FDC`, where:

`nnnnn` is the process id reporting the error
`mm` is a sequence number, normally 0

When a process creates an FFST it also sends a record to syslog. The record contains the name of the FFST file to assist in automatic problem tracking.

The syslog entry is made at the “user.error” level.

The MQSeries trace utility is explained in detail in the *MQSeries System Administration* (MQSeries Version 5 products) and the relevant *System Management Guide* for other platforms.

Using trace on VM/ESA

Use the MQTRACE environment variable to set the tracing on (see “MQSERVER” on page 114). Specify the name of the file to which you want all the trace entries to be put. Unlike MQTRACE for DOS and Windows, there are no optional flags that can be set. Setting the MQTRACE variable initiates a trace for all the functions of the MQSeries client for VM/ESA.

Example VM/ESA trace data

The following example shows an extract of a trace for a VM/ESA MQSeries client:

```
MQSeries Trace started at 10/09/97 16:07:14
< xcsInitialize (rc = OK)
-> MQCONN
--> rrxOpenChannelDef
---> xcsGetMem
<-- xcsGetMem (rc = OK)
<- rrxOpenChannelDef (rc = OK)
--> rrxGetFirstChannelDef
<- rrxGetFirstChannelDef (rc = OK)
--> rriInitSess
---> xcsGetMem
<-- xcsGetMem (rc = OK)
...

...
---> rriTermExits
<-- rriTermExits (rc = OK)
---> rriDeleteStatusEntry
----> xcsFreeMem
<--- xcsFreeMem (rc = OK)
<-- rriDeleteStatusEntry (rc = OK)
---> xcsFreeMem
<-- xcsFreeMem (rc = OK)
<- rriFreeSess (rc = OK)
< MQDISC
```

Figure 3. Extract from a VM/ESA client trace

Part 4. Appendix

Appendix. Notices

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	AS/400	BookManager
CICS	ESCON	FFST
First Failure Support Technology	IBM	IBMLink
IMS	MQ	MQSeries
MQSeries Three Tier	MVS	MVS/ESA
NetView	PS/2	RACF
RISC System/6000	RS/6000	SupportPac
OS/2	OS/400	VisualAge
VM/ESA	VSE/ESA	WIN-OS/2
Workplace		

Part 5. Glossary and index

Glossary of terms and abbreviations

This glossary defines MQSeries terms and abbreviations used in this book. If you do not find the term you are looking for, see the Index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

A

adapter. An interface between MQSeries for MVS/ESA and TSO, IMS, CICS, or batch address spaces. An adapter is an attachment facility that enables applications to access MQSeries services.

asynchronous messaging. A method of communication between programs in which programs place messages on message queues. With asynchronous messaging, the sending program proceeds with its own processing without waiting for a reply to its message. Contrast with *synchronous messaging*.

attribute. One of a set of properties that defines the characteristics of an MQSeries object.

C

CCSID. Coded character set identifier.

CDF. Channel definition file.

channel. See *message channel*.

channel definition file (CDF). In MQSeries, a file containing communication channel definitions that associate transmission queues with communication links.

CL. Control Language.

client. A run-time component that provides access to queuing services on a server for local user applications. The queues used by the applications reside on the server. See also *MQSeries client*.

client application. An application, running on a workstation and linked to a client, that gives the application access to queuing services on a server.

client connection channel type. The type of MQI channel definition associated with an MQSeries client. See also *server connection channel type*.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

command. In MQSeries, an instruction that can be carried out by the queue manager.

completion code. A return code indicating how an MQI call has ended.

configuration file. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, a file that contains configuration information related to, for example, logs, communications, or installable services. Synonymous with *.ini file*. See also *stanza*.

connect. To provide a queue manager connection handle, which an application uses on subsequent MQI calls. The connection is made either by the MQCONN call, or automatically by the MQOPEN call.

connection handle. The identifier or token by which a program accesses the queue manager to which it is connected.

Control Language (CL). In MQSeries for AS/400, a language that can be used to issue commands, either at the command line or by writing a CL program.

D

DCE. Distributed Computing Environment.

desktop clients. A group of MQSeries clients that run on the smaller platforms such as DOS and Windows 3.1. Desktop clients are supplied with most of the MQSeries products; the members of the group may be different for the different products.

Distributed Computing Environment (DCE). Middleware that provides some basic services, making the development of distributed applications easier. DCE is defined by the Open Software Foundation (OSF).

E

environment variable. One of a series of variables that control the way your operating system runs and what external devices it will recognize. You can define these variables in your system profile or override them temporarily with command-line commands.

F

FFST. First Failure Support Technology.

First Failure Support Technology (FFST). Used by MQSeries on UNIX systems, MQSeries for OS/2 Warp, MQSeries for Windows NT, and MQSeries for AS/400 to detect and report software problems.

G

get. In message queuing, to use the MQGET call to remove a message from a queue.

H

handle. See *connection handle* and *object handle*.

I

.ini file. See *configuration file*.

initiation queue. A local queue on which the queue manager puts trigger messages.

L

listener. In MQSeries distributed queuing, a program that monitors for incoming network connections.

local queue. A queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with *remote queue*.

local queue manager. The queue manager to which a program is connected and that provides message queuing services to the program. Queue managers to which a program is not connected are called *remote queue managers*, even if they are running on the same system as the program.

log. In MQSeries, a file recording the work done by queue managers while they receive, transmit, and deliver messages.

log file. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, a file in which all significant changes to the data controlled by a queue manager are recorded. If the primary log files become full, MQSeries allocates secondary log files.

M

MCA. Message channel agent.

message. (1) In message queuing applications, a communication sent between programs. See also *persistent message* and *nonpersistent message*. (2) In system programming, information intended for the terminal operator or system administrator.

message channel. In distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender and a receiver) and a communication link. Contrast with *MQI channel*.

message channel agent (MCA). A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue.

message queue. Synonym for *queue*.

message queue interface (MQI). The programming interface provided by the MQSeries queue managers. This programming interface allows application programs to access message queuing services.

message queuing. A programming technique in which each program within an application communicates with the other programs by putting messages on queues.

messaging. See *synchronous messaging* and *asynchronous messaging*.

MQI. Message queue interface.

MQI channel. Connects an MQSeries client to a queue manager on a server system, and transfers only MQI calls and responses in a bidirectional manner. Contrast with *message channel*.

MQSC. MQSeries commands.

MQSeries. A family of IBM licensed programs that provides message queuing services.

MQSeries client. Part of an MQSeries product that can be installed on a system without installing the full queue manager. The MQSeries client accepts MQI calls from applications and communicates with a queue manager on a server system.

MQSeries commands (MQSC). Human readable commands, uniform across all platforms, that are used to manipulate MQSeries objects. Contrast with *programmable command format (PCF)*.

N

namelist. An MQSeries for MVS/ESA object that contains a list of queue names.

O

object. In MQSeries, an object is a queue manager, a queue, a process definition, a channel, a namelist (MVS/ESA only), or a storage class (MVS/ESA only).

object handle. The identifier or token by which a program accesses the MQSeries object with which it is working.

P

PCF. Programmable command format.

PCF command. See *programmable command format*.

ping. In distributed queuing, a diagnostic aid that uses the exchange of a test message to confirm that a message channel or a TCP/IP connection is functioning.

platform. In MQSeries, the operating system under which a queue manager is running.

programmable command format (PCF). A type of MQSeries message used by:

- User administration applications, to put PCF commands onto the system command input queue of a specified queue manager
- User administration applications, to get the results of a PCF command from a specified queue manager
- A queue manager, as a notification that an event has occurred

Contrast with *MQSC*.

Q

queue. An MQSeries object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages—they point to other queues, or can be used as models for dynamic queues.

queue manager. (1) A system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. See also *local queue manager* and *remote queue manager*. (2) An MQSeries object that defines the attributes of a particular queue manager.

queuing. See *message queuing*.

R

reason code. A return code that describes the reason for the failure or partial success of an MQI call.

receiver channel. In message queuing, a channel that responds to a sender channel, takes messages from a communication link, and puts them on a local queue.

Registry. In Windows NT, a secure database that provides a single source for system and application configuration data.

Registry Editor. In Windows NT, the program item that allows the user to edit the Registry.

Registry Hive. In Windows NT, the structure of the data stored in the Registry.

remote queue. A queue belonging to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with *local queue*.

remote queue manager. To a program, a queue manager that is not the one to which the program is connected.

remote queuing. In message queuing, the provision of services to enable applications to put messages on queues belonging to other queue managers.

resource. Any facility of the computing system or operating system required by a job or task. In MQSeries for MVS/ESA, examples of resources are buffer pools, page sets, log data sets, queues, and messages.

return codes. The collective name for completion codes and reason codes.

S

server. (1) In MQSeries, a queue manager that provides queue services to client applications running on a remote workstation. (2) The program that responds to requests for information in the particular two-program, information-flow model of client/server. See also *client*.

server channel. In message queuing, a channel that responds to a requester channel, removes messages from a transmission queue, and moves them over a communication link to the requester channel.

server connection channel type. The type of MQI channel definition associated with the server that runs a queue manager. See also *client connection channel type*.

stanza. A group of lines in a configuration file that assigns a value to a parameter modifying the behavior of a queue manager, client, or channel. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, a configuration (.ini) file may contain a number of stanzas.

synchronous messaging. A method of communication between programs in which programs place messages on message queues. With synchronous messaging, the sending program waits for a reply to its message before resuming its own processing. Contrast with *asynchronous messaging*.

syncpoint. An intermediate or end point during processing of a transaction at which the transaction's protected resources are consistent. At a syncpoint, changes to the resources can safely be committed, or they can be backed out to the previous syncpoint.

T

thread. In MQSeries, the lowest level of parallel execution available on an operating system platform.

time-independent messaging. See *asynchronous messaging*.

trace. In MQSeries, a facility for recording MQSeries activity. The destinations for trace entries can include GTF and the system management facility (SMF).

triggering. In MQSeries, a facility allowing a queue manager to start an application automatically when predetermined conditions on a queue are satisfied.

trigger monitor. A continuously-running application serving one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

U

utility. In MQSeries, a supplied set of programs that provide the system operator or system administrator with facilities in addition to those provided by the MQSeries commands. Some utilities invoke more than one function.

Index

Sending your comments to IBM

MQSeries

Clients

GC33-1632-05

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, use the Readers' Comment Form
- 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: WINVMD(IDRCF)
  - Internet: idrcf@winvmd.vnet.ibm.com

Whichever you use, ensure that you include:

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

Readers' Comments

MQSeries

Clients

GC33-1632-05

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

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. This form is provided for comments about the information in this manual and the way it is presented.

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.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

MQSeries

MQSeries Clients GC33-1632-05

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States of America
Belgium	France	Israel	New Zealand	Spain	
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

1 Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories
Information Development Department (MP095)
Hursley Park,
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name ______________________

Company or Organization ______________________

Address ______________________

____

EMAIL ______________________

Telephone ______________________

1 Cut along this line

4 Fasten here with adhesive tape

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

GC33-1632-05

Spine information:

IBM MQSeries Clients