MQSeries®

Clients

> **Note!**
>
> Before using this information and the product it supports, be sure to read the general information under "Appendix C. Notices" on page 175.

Tenth edition (November 2000)

This edition applies to the following products:
- MQSeries for AIX, V5.1
- MQSeries for AS/400® Version 5.1
- MQSeries for AT&T GIS UNIX® Version 2 Release 2
- MQSeries for Compaq (DIGITAL) OpenVMS Version 2 Release 2
- MQSeries for Compaq Tru64 UNIX Version 5 Release 1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V2.1
- MQSeries for OS/390 V5.2
- MQSeries for SINIX and DC/OSx Version 2 Release 2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Sun Solaris, Intel Platform Edition Version 5 Release 1
- MQSeries for Tandem NonStop Kernel Version 2 Release 2.0.1
- MQSeries for VSE/ESA™ Version 2 Release 1
- MQSeries for Windows NT, V5.1
- MQSeries for Windows® Version 2 Release 0
- MQSeries for Windows Version 2 Release 1
- MQSeries link for R/3 for AIX® Version 1.2
- MQSeries link for R/3 for AS/400 Version 1.2
- MQSeries link for R/3 for Compaq Tru64 UNIX Version 1.2
- MQSeries link for R/3 for HP-UX Version 1.2
- MQSeries link for R/3 for Sun Solaris Version 1.2
- MQSeries link for R/3 for Windows NT® Version 1.2

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

Contents

Figures

Tables

About this book

This book contains information about the MQSeries client/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).

What you need to know to understand this book

To understand this book, you should have:
- Experience in installing and configuring the system you use for the server. This can be AS/400, MQSeries for Compaq (DIGITAL) OpenVMS, OS/2 Warp, OS/390, Tandem NonStop Kernel (NSK), one of the UNIX systems listed below, VSE/ESA, or Windows NT.
- Experience with any client platforms that you will be using, for example, DOS, Windows 3.1, Windows 95, and Windows 98.
- Understanding of the purpose of the Message Queue Interface (MQI).

  For further information on MQI, see the following manuals:

  – For MQI call constants, see the *MQSeries Application Programming Reference* manual.

  – For events, commands, and responses, see the *MQSeries Programmable System Management* manual.
- 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 of MQSeries clients" on page 3 first as a brief introduction. There you will see "How do I set up an MQSeries client?" on page 5 which gives you a list of tasks that you might need to carry out; this guides you through the rest of the book.

Terms used in this book

In this book, references to "UNIX® systems" include:
- AIX
- AT&T GIS UNIX (this platform has become NCR UNIX)
- DIGITAL UNIX (Compaq Tru64 UNIX)
- HP-UX

About this book

- SINIX and DC/OSx
- Sun Solaris (SPARC and Intel Platform Editions)

References to MQSeries for UNIX systems include:
- IBM MQSeries for AIX Version 5.1
- IBM MQSeries for AT&T GIS UNIX Version 2.2
- IBM MQSeries for Compaq Tru64 UNIX
- IBM MQSeries for HP-UX Version 5.1
- IBM MQSeries for SINIX and DC/OSx Version 2.2
- IBM MQSeries for Sun Solaris Version 5.1
- IBM MQSeries for Sun Solaris, Intel Platform Edition Version 5 Release 1

Throughout this book, the name *mqmtop* has been used to represent the name of the base directory where MQSeries is installed on UNIX systems.
- For AIX, the name of the actual directory is **/usr/mqm**
- For other UNIX systems, the name of the actual directory is **/opt/mqm**

Summary of changes

This section describes changes in this edition of *MQSeries Clients*. Changes since the previous edition of the book are marked by vertical lines to the left of the changes.

Changes for this edition (GC33-1632-09)

The following products have been added:
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for Sun Solaris, Intel Platform Edition, V5.1

The following sections have been extended to include information on queue-sharing groups:
- "Chapter 8. Using channels" on page 119
- "Chapter 12. Running applications on MQSeries clients" on page 151

A section on connecting a client to a queue manager in a queue-sharing group has been added.

Changes for the ninth edition (GC33-1632-08)

This edition was not published.

Changes for the eighth edition

The following products were added:
- MQSeries for AS/400, V5.1
- MQSeries for Compaq Tru64 UNIX Version 5 Release 1
- MQSeries for Tandem NonStop Kernel, V2.2.0.1

Changes for the seventh edition

The following products were added:
- MQSeries for AIX V5.1
- MQSeries for AS/400 V4R2M1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2® Warp V5.1
- MQSeries for OS/390® V2.1
- MQSeries for Sun Solaris V5.1
- MQSeries for VSE/ESA V2.1
- MQSeries for Windows NT V5.1

Changes

Part 1. Introduction to MQSeries clients

1

Introduction

Chapter 1. Overview of MQSeries clients

This chapter discusses the following topics:
- "What is an MQSeries client?"
- "Why use MQSeries clients?" on page 5
- "How do I set up an MQSeries client?" on page 5

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 with one or more MQSeries servers and connect to their queue managers by means of a communications protocol. The servers to which the client connects might or might not be part of a cluster.
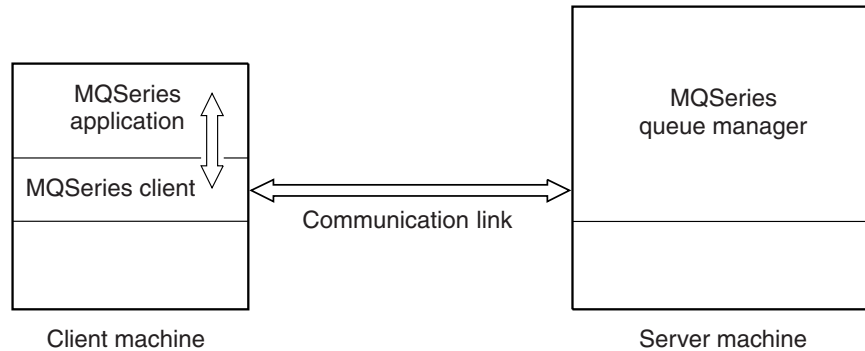
Figure 1. Link between a client and server

The following platforms can be used. The combinations depend on which MQSeries product you are using and are described in "Platform support for MQSeries clients" on page 11. Other MQSeries clients are also available; these are described in "MQSeries clients on other platforms" on page 12.

MQSeries client	MQSeries server
Compaq (DIGITAL) OpenVMS	AS/400
DOS	Compaq (DIGITAL) OpenVMS
OS/2 Warp	MVS/ESA
UNIX systems	OS/2 Warp
VM/ESA	OS/390
Windows NT	Tandem NSK
Windows 3.1	UNIX systems
Windows 95	VSE/ESA
Windows 98	Windows NT

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.

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 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 is established dynamically at runtime (except in the case of DOS, when it is a static link).

How the client connects to the server

An application running in the MQSeries client environment runs in synchronous mode because 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. Clients and servers communicate through *MQI channels*. 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 when your machine also 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 program 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. For example, an MQSeries client
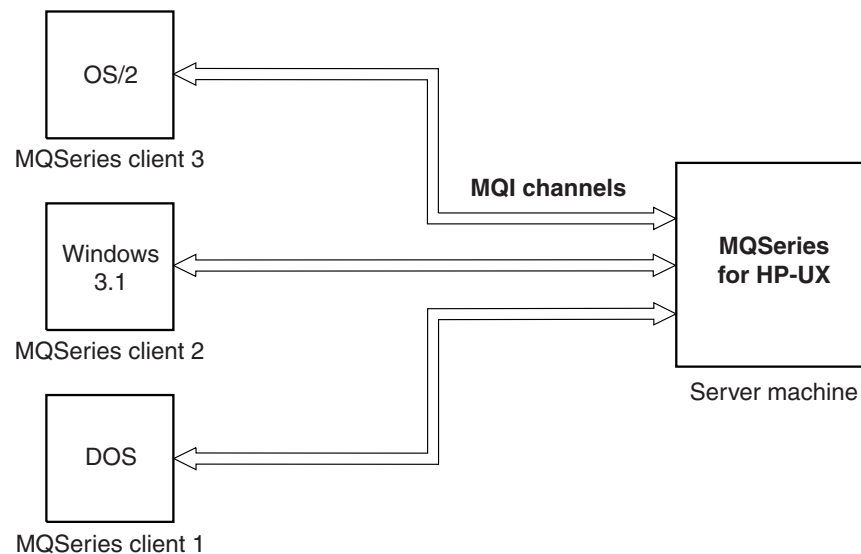
Figure 2. MQSeries server connected to clients on different platforms

can connect to more than one queue manager, or any number of queue managers connected as part of a queue-sharing group.

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, Windows 3.1, Windows 95, or Windows 98 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 can 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
- **MQGET** with signal

An application running on an MQSeries client can connect to more than one queue manager concurrently, or use a queue manager name with an asterisk (*) on an **MQCONN** or **MQCONNX** call (see the examples in "Chapter 12. Running applications on MQSeries clients" on page 151). The **MQCONNX** call is only supported in client libraries.

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 a machine, to which your client will connect. The steps involved in setting up a client are:

1. Check that you have a suitable platform for an MQSeries client and that the hardware and software satisfy the requirements. This is described in "Chapter 2. Preparing for installation" on page 11.
2. 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. This is described in "Chapter 3. Installing MQSeries client components from Version 5 products (not OS/390)" on page 29, and "Chapter 4. Installing MQSeries clients with non-Version 5 products, and OS/390" on page 71.
3. Ensure that your communication links are configured and connected. This is described in "Chapter 5. Configuring communication links" on page 83.
4. Check that your installation is working correctly. This is described in "Chapter 6. Verifying the installation" on page 105.
5. When you have the verified MQSeries client installation, consider whether you need to take any action on security. This is described in "Chapter 7. Setting up MQSeries client security" on page 115.

Overview of MQSeries clients

6. Set up the channels between the MQSeries client and server that are required by the MQSeries applications you want to run on the client. This is described in "Chapter 8. Using channels" on page 119. You might need to use MQSeries *environment variables* to set up the channels. These are described in "Chapter 9. Using MQSeries environment variables" on page 131.

7. MQSeries applications are fully described in the *MQSeries Application Programming Guide*.

8. There are some differences to consider when designing, building and running applications in the MQSeries client environment. For information about these differences, see:
   - "Chapter 10. Using the message queue interface (MQI)" on page 139
   - "Chapter 11. Building applications for MQSeries clients" on page 143
   - "Chapter 12. Running applications on MQSeries clients" on page 151
   - "Chapter 13. Solving problems" on page 157

Part 2. Installing MQSeries clients

Installing MQSeries clients

Installing MQSeries clients

Chapter 2. Preparing for installation

This chapter discusses the following topics:
- "Platform support for MQSeries clients"
- "Communications" on page 12
- "Hardware and software requirements" on page 14

Platform 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 Client Attachment feature* on MQSeries for OS/390). 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.

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* book for more information.

The following MQSeries products:
- **Version 5 products**:
  - MQSeries for AIX, V5.1
  - MQSeries for AS/400, V5.1
  - MQSeries for Compaq Tru64 UNIX, V5.1
  - MQSeries for HP-UX, V5.1
  - MQSeries for OS/2 Warp, V5.1
  - MQSeries for OS/390 Version 5 Release 2
  - MQSeries for Sun Solaris, Intel Platform Edition, V5.1
  - MQSeries for Windows NT, V5.1
- **Non-Version 5 products**:
  - MQSeries for AT&T GIS UNIX V2.2
  - MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS Version 2.2
  - MQSeries for SINIX and DC/OSx, V2.2
  - MQSeries for Tandem NonStop Kernel, V2.2.0.1
  - MQSeries for MVS/ESA™ Version 1 Release 2
  - MQSeries for VSE/ESA Version 2 Release 1

can accept connections from MQSeries clients on the following platforms:
- AIX
- AT&T GIS UNIX (this platform has become NCR UNIX)
- Compaq Tru64 UNIX
- DOS
- HP-UX
- OS/2 Warp
- SINIX and DC/OSx
- Sun Solaris (SPARC and Intel Platform Editions)
- VM/ESA®®
- Windows 3.1
- Windows 95
- Windows 98
- Windows NT

Platform support for MQSeries clients

MQSeries for Windows Version 2.0 and 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*.

Applications on Version 5 clients

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

MQSeries clients on other platforms

Most MQSeries products supply files for clients on the same platform as the server and for clients on other platforms. The following MQSeries products supply files for clients on other platforms only:

- MQSeries for AS/400
- MQSeries for MVS/ESA
- MQSeries for OS/390
- MQSeries for Tandem NSK
- MQSeries for VSE/ESA

For details see "MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS, and UNIX platforms" on page 73 and "Chapter 3. Installing MQSeries client components from Version 5 products (not OS/390)" on page 29.

Further MQSeries Clients are available through the Internet as SupportPacs. See "MQSeries information available on the Internet" on page 185.

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 "Defining MQI channels" on page 120.

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
MQSeries for Compaq (DIGITAL) OpenVMS	✔	✔			✔
DOS		✔	✔	✔	
OS/2 Warp	✔	✔	✔	✔	
UNIX platforms	✔(1)	✔			
VM/ESA	✔	✔			
Windows 3.1		✔	✔	✔	
Windows 95		✔	✔	✔	
Windows 98		✔	✔	✔	

Table 1. Transmission protocols for MQI channels (continued)

Client platform	LU 6.2	TCP/IP	NetBIOS	SPX	DECnet
Windows NT	✔	✔	✔	✔	
Note:					
1. Except on Sun Solaris, Intel Platform Edition					

Table 2 on page 84 shows 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-independent and time-independent.

The transmission protocol is specified via the channel definition and does not affect the application. For example, a 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 might affect the performance of the MQSeries client and server system.

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

Hardware and software requirements

The following table shows where you can find hardware and software requirements for the client platforms.

Platform	Page
AIX	14
AT&T GIS UNIX (NCR UNIX)	16
MQSeries for Compaq (DIGITAL) OpenVMS	17
Compaq Tru64 UNIX	17
DOS	18
HP-UX	20
OS/2 Warp	21
SINIX and DC/OSx	22
VM/ESA	24
Sun Solaris	23
Sun Solaris, Intel Platform Edition	24
Windows 3.1	26
Windows 95 and Windows 98	27
Windows NT	28

For your server platform hardware and software requirements, see the manual that describes installation for your platform.

AIX client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for AIX.

Hardware
An MQSeries client can run on an IBM RISC System/6000®, capable of running AIX Version 4.2 or later. Any other trademarked AIX system can 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, the access methods, and the application programs.

Software
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.
- AIX Version 4.2 (5765-655 or 5765-C34) or AIX Version 4.3 or later.

  Later levels of some listed products might be required for AIX Version 4.2, SMP, and SP™. Later levels of operating system might be required to support corequisite products.

Connectivity: For TCP/IP connectivity:
- TCP/IP (in the operating system)

For SNA LU 6.2 connectivity:
- IBM eNetwork™ Communications Server for AIX, Version 5.0

Optional software:

- IBM DCE Version 2.2. This must be a DCE product which supports DES data 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 compilers are supported:
- IBM C for AIX, Version 4.4
- IBM C Set++ for AIX, Version 3.1 (5765-421)
- IBM C and C++ compiler, Version 3.6
- IBM VisualAge C++ Professional for AIX, Version 4.0
- IBM COBOL Set for AIX, Version 1.1
- Micro Focus COBOL Compiler for UNIX, Version 4.0
- IBM PL/I Set for AIX, Version 1.1
- IBM VisualAge Java Enterprise Edition for AIX, Version 2.0

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

This section outlines the hardware and software requirements for an MQSeries client for AT&T GIS UNIX (NCR UNIX).

Hardware
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 or TCP/IP

Software
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.
- AT&T GIS UNIX SVR4 MP-RAS, Version 3.0, including TCP/IP (this platform has become NCR UNIX SVR4 MP-RAS, R3.0)

Connectivity:
- AT&T GIS SNA Services, Version 2.06 or later Version 2
- TCP/IP as part of the base operating system

Optional software: None.

Compiler for MQSeries applications on AT&T GIS UNIX clients
The following C compiler is supported:
- AT&T GIS High Performance C, Version 1.0b

Compaq Tru64 UNIX client: hardware and software required

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

Hardware

An MQSeries client can run only on:
- Any desktop or server system capable of running Compaq Tru64 UNIX

with a minimum system disk space of 25 MB.

For connectivity, any communications hardware supporting TCP/IP.

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

Software

The following are prerequisites for MQSeries applications running on a Compaq Tru64 UNIX client.
- Compaq Tru64 UNIX Version 4.0 or Version 5.0

Connectivity

- TCP/IP as part of the base operating system

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

Compilers for MQSeries applications on Compaq Tru64 UNIX clients

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

Digital OpenVMS client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for Digital OpenVMS.

Hardware

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.

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

Software

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

---

1. Supplied by Compaq

2. Supplied by Micro Focus

Hardware and software, Digital OpenVMS

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

- OpenVMS operating system, Version 6.2

Connectivity:
- Digital SNA APPC LU 6.2 Programming Interface, Version 2.3
- Digital DECnet SNA Gateway software, Version 1.2A
- Process Software TCPWare, Version 5.2-3
- 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), Version 4.1
- VAX: TCP/IP Services for OpenVMS VAX, Version 3.3
- Attachmate Pathway for OpenVMS, Version 2.5.1

Optional software:
- DCE
  - Distributed Computing Environment for OpenVMS, Version 1.3b

Compilers for MQSeries applications on Digital OpenVMS clients
The following compilers are supported:
- AXP/VAX: DEC C, Version 5.2
- AXP: DEC C++, Version 5.2
- VAX: DEC C++, Version 5.0
- VAX: VAX COBOL, Version 5.3
- AXP: DEC COBOL, Version 2.2

DOS client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for DOS.

Hardware
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, the access methods, and the application programs.

Software
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.

- DOS, Version 5.0 or later

Connectivity:
- TCP/IP (in the operating system)
- SPX

Optional software:
- Novell Netware Client for DOS/Win, Version 1.20 and Version 2.5
- Novell LAN Workplace, Version 5.1
- FTP PC/TCP for DOS, Version 5.0

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

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

Compilers for MQSeries applications on DOS clients

The following compiler is supported:

* Microsoft Visual C++, Version 1.5

HP-UX client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for HP-UX.

Hardware

An MQSeries client can run on HP-UX on any HP 9000 Series 700 or Series 800 or Stratus Continuum/400 machine, with minimum system disk space of 20 MB.

Software

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.
- HP-UX, Version 10.20
- HP-UX, Version 11

Connectivity: For TCP/IP connectivity:

- TCP/IP (in the operating system)

For SNA connectivity:

- HP SNAplusII

Optional software:

- The HP DCE/9000 version appropriate for the level of the HP-UX operating system in use.
- 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 compilers are supported:
- Micro Focus COBOL for UNIX, Version 4.0
- HP-UX ANSI C compiler
- C Softbench, Version 5.0
- HP CFRONT C++, Version 3.1
- IBM C and C++ compilers, Version 3.6
- HP ANSI C++

OS/2 Warp client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for OS/2 Warp.

Hardware

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, the access methods, and the application programs.

The system unit must have a CD-ROM device.

Software

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

This is the minimum supported software level. Later levels, if any, will be supported unless otherwise stated.
- OS/2 Warp, Version 4.0 (84H1426) (5622-851)
- OS/2 Warp Server, Version 4.0 (25H8002)
- OS/2 Warp Server Advanced SMP feature, Version 4.0
- OS/2 Workspace-on-Demand

Connectivity:
- IBM eNetwork Communications Server for OS/2 Warp, Version 5.0
- Novell Netware Client for OS/2 Warp, Version 1.20 (for direct IPX/SPX support)
- IntraNetWare Client for OS/2 Warp, Version 2.12
- TCP/IP for OS/2 Warp, Version 2.0 base kit plus NetBIOS kit, Version 3.5 (base kit is essential)
- NetWare for OS/2 Warp, Version 4.11

Optional software:
- IBM Directory and Security Server for OS/2 Warp, Version 4 or later compatible versions. This must be a DCE product which supports DES data 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, Version 4.0 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 Warp clients

The following compilers are supported:
- IBM VisualAge® COBOL for OS/2 Warp, Version 1.1 and Version 2.2
- Micro Focus COBOL compiler, Version 4.0 (32 bit)
- IBM VisualAge C++ for OS/2 Warp, Version 3.0
- Borland C++ compiler, Version 2.0 and Version 5.02 (C bindings only)
- IBM C and C++ compilers, Version 3.6
- IBM PL/I for OS/2 Warp, Version 1.2
- IBM VisualAge for PL/I for OS/2 Warp
- IBM VisualAge Java Enterprise Edition for OS/2 Warp, Version 2.0
- IBM VisualAge Java Professional Edition for OS/2 Warp, Version 2.0

SINIX and DC/OSx client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for SINIX and DC/OSx.

Hardware

An MQSeries client can run on:

- SINIX: RM200, RM300, RM400, RM600 systems with minimum system disk space of 30 MB.
- DC/OSx: MIServer, Nile systems with minimum system disk space of 30 MB.
- Any communications hardware supporting SNA LU 6.2 or TCP/IP.

Software

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

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, Version 1.2
      - ISC with SNA engine, Version 1.3
    - To support the ILC-T (Intelligent LAN Controller, Token ring) interface:
      - Comm Services, Version 1.2
      - Token Ring Mac interface, Version 1.3
    - To support the SNA on the ESCON® IBM Channel link:
      - XVI/ESCON Driver 1.0
- DCE
    - SINIX: Version 1.03A00 or later

Optional software: None.

Compilers for MQSeries applications on SINIX and DC/OSx clients

The following compilers are supported:

- SINIX: C compiler (C-DS, MIPS), Version 1.1
- DC/OSx: C4.0 compiler, Version 4.0.1
- SINIX: Micro Focus COBOL, Version 3.2
- DC/OSx: Micro Focus COBOL, Version 3.2

Sun Solaris client: hardware and software required

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

Hardware
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.

Note: Solaris systems from other manufacturers are not supported.

For connectivity, any communications hardware supporting SNA LU 6.2 or TCP/IP.

Software
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, Version 2.6 with patches 105210-13 and 105568-10, and Sun Solaris 7.

Connectivity:
- SunLink SNA Peer-to-Peer, Version 9.1
- TCP/IP as part of the base operating system
- If token ring is to be used: SunLink Token Ring Interface /SBus, Version 3.0.2. This requires patch 102463 or Sun TRI/P Adapter, Version 1.0

Optional software: Transarc DCE, Version 1.1. This must be a DCE product which supports DES data encryption if the user wishes to run the MQSeries-supplied DCE send, receive, or message exits.

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

Compilers for MQSeries applications on Sun Solaris clients
The following compilers are supported:
- SunWorkShop Compiler C, Version 4.2
- SunWorkShop Compiler C++, Version 4.2

Sun Solaris, Intel Client: Hardware and Software Required

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

Hardware

An MQSeries client can run only on:
- Sun Intel desktop or server

with a minimum system disk space of 25 MB.

Note: Solaris systems from other manufacturers are not supported.

For connectivity, any communications hardware supporting TCP/IP.

Software

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

IBM® MQSeries for Sun Solaris, Intel, Version 5.1 clients run on:
- Sun Solaris 7 Intel Platform Edition Hardware 11/99, with patches 106981-10, 106542-10, 107637-03, and 107172-05

Connectivity
- TCP/IP as part of the base operating system

Compilers for MQSeries Applications on Sun Solaris, Intel Clients

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

VM/ESA client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for VM/ESA.

Hardware

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

Software

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

Connectivity:
- TCP/IP, Release 2.4
- VTAM® LU 6.2

Optional software: None.

Compilers for MQSeries applications on VM/ESA clients

The following compilers are supported:

- IBM Assembler
- IBM VS COBOL II
- IBM C for VM, Release 3.1
- IBM OS/PL/I, Release 2.3
- IBM VM/ESA Rexx/VM

Windows 3.1 client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for Windows 3.1.

Hardware

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, the access methods, and the application programs.

Software

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.
- Windows 3.1
- Windows 95 in 16-bit mode
- Windows for Workgroups

Connectivity:
- TCP/IP
- SPX

Optional software:
- Novell Netware client for DOS/Win31, Version 1.20
- SunPC NSF, Version 5.1
- OnNet SDK for Windows

Compilers for MQSeries applications on Windows 3.1 clients

The following compiler is supported:
- Microsoft Visual C++, Version 1.5

Windows 95 and Windows 98 client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for Windows 95 and Windows 98.

Hardware

An MQSeries client can run on Windows 95 or Windows 98 on a personal computer. There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the MQSeries client, the access methods, and the application programs.

Software

The following are prerequisites for MQSeries applications running on a Windows 95 and Windows 98 client.

Minimum supported software levels are shown. Later levels, if any, will be supported unless otherwise stated.
- Windows 95 or Windows 98

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

Optional software:
- IBM DCE for Windows 95, Version 1.1

  A DCE security module is provided as part of the MQSeries client for Windows 95 and Windows 98. IBM DCE for Windows 98 does not currently support DES encryption, so you cannot run the DCE send, receive, or message exits supplied by MQSeries.

Compilers for MQSeries applications on Windows 95 and Windows 98 clients

The following compilers are supported:
- Micro Focus COBOL Workbench, Version 4.0
- IBM VisualAge C++ for Windows, Version 3.5
- Microsoft Visual C++ for Windows 95/NT, Version 4.0 and Version 5.0

Windows NT client: hardware and software required

This section outlines the hardware and software requirements for an MQSeries client for Windows NT.

Hardware

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

Software

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.

- Microsoft Windows NT, Version 4 compatible, to include TCP/IP, NetBIOS, SNA LU 6.2, and SPX.

Connectivity:
- IBM Communications Server for Windows NT, Version 5.0
- Attachmate Extra! Personal Client, Version 6.1 and Version 6.2
- Microsoft SNA Server, Version 2.11 or Version 3
- TCP/IP, NetBIOS, and SPX are part of the base operating system

Optional software:
- IBM DCE, Version 1.1

Compilers for MQSeries applications on Windows NT clients

The following compilers are supported:
- IBM VisualAge COBOL for Windows NT, Version 2.1
- IBM VisualAge COBOL Enterprise, Version 2.2
- IBM VisualAge C++, Version 3.5
- Microsoft Visual C++ for Windows 95 and Windows NT, Version 4.0 and Version 5.0
- IBM C and C++ compilers, Version 3.6
- IBM PL/I for Windows, Version 1.2
- IBM VisualAge for PL/I for Windows
- IBM VisualAge PL/I Enterprise, Version 2.1
- IBM Visual Basic for Windows, Version 4.0 (16-bit) and Version 5.0 (32-bit) or higher
- IBM VisualAge for Java, e-Business Edition for Windows 95 and Windows NT
- IBM VisualAge Java Enterprise Edition, Version 2.0
- IBM VisualAge Java Professional Edition, Version 2.0

Chapter 3. Installing MQSeries client components from Version 5 products (not OS/390)

This chapter discusses how to install the client components for the following Version 5 MQSeries products:
- MQSeries for AIX, V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for Sun Solaris, V5.1
- MQSeries for Sun Solaris, Intel Platform Edition, V5.1
- MQSeries for Windows NT, V5.1

For information on installing MQSeries client components from MQSeries for OS/390, see "Chapter 4. Installing MQSeries clients with non-Version 5 products, and OS/390" on page 71.

MQSeries Version 5 products (not OS/390) 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, and OS/390" on page 71.

Each MQSeries Version 5 product (not OS/390) supplies software, including the easy installation feature, for clients on the following platforms:
- AIX
- DOS
- HP-UX
- OS/2 Warp
- Sun Solaris
- Windows 3.1
- Windows 95 and Windows 98
- Windows NT

Further MQSeries clients are available through the Internet, as described in "Obtaining MQSeries clients from IBM Transaction Processing SupportPacs" on page 72 and "MQSeries information available on the Internet" on page 185.

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.

Note: The MQSeries for Compaq Tru64 UNIX server CD-Rom contains the client components for that platform.

See the step by step instructions for your client platform:
- "Installing on AIX" on page 31

Installing clients (Version 5)

It is possible to install both the client and server components using the MQSeries Server CD-ROM, as explained in "Installing MQSeries clients on the same machine as the server".

Installing MQSeries clients on the same machine as the server

To install MQSeries client components on an MQSeries server machine, use the MQSeries Server CD-ROM (not the MQSeries Client CD-ROM). Choose the client install option to install client components from the MQSeries Server CD-ROM.

Use the MQSeries Client CD-ROM only to install MQSeries client components on a machine that is not an MQSeries server.

You may install client components from the MQSeries Client CD-ROM and later decide to install the MQSeries server on the same machine. Before you can do this, you must 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.

Remember that even when your server and client both reside on the same machine, you still need to define the MQI channels between them. See "Chapter 8. Using channels" on page 119 for details.

Uninstalling MQSeries clients

If you want to remove the MQSeries client files from your system, use the process provided to do this efficiently. Details are given after the installation instructions for each platform.

Installing on AIX

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

The MQSeries client is installed into the **/usr/mqm** directory. This *cannot* be changed. However, if you do not have enough space in the **/usr/mqm** file system, follow the procedure given in "Creating another file system for the client" on page 32.

If you have a previous version of the MQSeries for AIX client installed on your system, or if a file system remains from a previous AIX client installation, see "Migrating from an earlier version of MQSeries for AIX" on page 34.

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 30.

Components for AIX

The components you can install on AIX 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 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 the *MQSeries Using Java*™ book.

MQSeries Internet Gateway documentation
> MQSeries Internet Gateway documentation 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.

Before installing

Before you can install the MQSeries client on your AIX system you are advised 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 filesets, the following directories *must* be on a local file system:
- /var/mqm
- /var/mqm/log

You can choose to NFS mount the /var/mqm/errors directory to conserve space on your local system.

A user ID with the name mqm, whose primary group is mqm, is automatically created during installation. You can create the user and group IDs yourself (see "Creating the mqm user ID and group"), but make sure you do this before installing the client. User ID and group must both be mqm.

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

After installation, the mqm user ID 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.

Creating another file system for the client

If you do not have enough space in the **/usr/mqm** file system, you can do either of the following things:

1. Create a new file system and mount it as **/usr/mqm**.

or

2. Create a new directory anywhere on your machine that is large enough to contain the client files, and create a symbolic link from **/usr/mqm** to this new directory. For example:

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

Whichever of these options you choose, you *must* do it before installing the client.

The file system into which the client is installed can be a remote network device, for example NFS, provided that the mount options are defined on that device to allow setuid programs (including root access) to be run.

Creating the mqm user ID and group

If you want to create the required IDs yourself, for instance, if you want to set up all security groups before installing client components, follow this procedure before you install the client. You must create both user ID and group as mqm.

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:

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

Set the name field to `mqm`

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

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

Set the name field to `mqm`

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:

```
Smit
   Security & Users
      Change a user's Password
```

If you are entering a new user `mqm`, the old password is `mqm`

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

```
Smit
   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 name field, add `mqm` to the **Group SET** field, which is a comma-separated list of the groups to which the user belongs.

Note: You need not have your primary group set to `mqm`. As long as `mqm` is in your set of groups, you can use the commands. If you are running applications that use the queue manager only, you do not need `mqm` group authority.

Easy installation

This section describes the 'Easy Installation' procedure using the MQSeries Client CD-ROM.

Note: If you have a previous version of the MQSeries for AIX client installed on your system, or if a file system remains from a previous AIX client installation, see "Migrating from an earlier version of MQSeries for AIX" on page 34.

1. Log on as root
2. Insert the MQSeries Client CD-ROM into the CD-ROM drive
3. Type `cd /cdrom` to change to the cdrom directory
4. Type `xinstallm -ez`

   Note: `xinstallm` is part of VSM (Visual System Management), which may not be installed on your system. It is part of the `xll.vsm.rte` fileset. You can use SMIT or the underlying installp command as alternatives

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
   ```
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 6. Verifying the installation" on page 105.

Custom installation

This section describes custom installation using the System Management Interface Tool (SMIT).

Note: If you have a previous version of the MQSeries for AIX client installed on your system, or if a file system remains from a previous AIX client installation, see "Migrating from an earlier version of MQSeries for AIX".

You can use SMIT for a custom installation as follows:

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

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

   ```
   Software Installation and Maintenance
      Install and Update Software
         Install and Update from Latest Available Software
   ```

   Alternatively, you can use the fastpath command:

   ```
   smitty install_latest
   ```
4. Press the **List** button to display the **Single Select List** window.
5. Select **/dev/cd0 (CD-ROM Drive)**
6. Press **Do** to display the parameters for **Install Latest Level**.
7. Press **F4** to get a list of filesets to install.
8. Follow the SMIT instructions to select the components you want to install.
9. Press **Enter**.
10. If you have a previous version of the product on your machine, change the **Automatically install requisite software** to **No** and **Overwrite same or newer versions** to **Yes**.
11. Press **OK** to install.

Now go to "Chapter 6. Verifying the installation" on page 105.

Migrating from an earlier version of MQSeries for AIX

If you want to migrate from an MQSeries for AIX V5.0 client to an MQSeries for AIX V5.1 client, you must first end all MQSeries activity on the target machine,

and remove any shared resources that are used by MQSeries. Do this either by shutting down the system and restarting it, or by issuing the **icprm** command to remove the shared resources.

The migration procedure described in this section applies only to migration from MQSeries for AIX V5.0 clients to MQSeries for AIX V5.1 clients. If you are migrating from an earlier version of MQSeries for AIX, you are advised to uninstall your current version before installing the MQSeries for AIX V5.1 client.

Migration from MQSeries for AIX V5.0 involves updating any currently installed filesets, and installing any new filesets that might be required.

To update currently installed filesets:

1. Go into SMIT for root authority. From the shell, enter:

   ```
   smit
   ```

2. Select the device appropriate for your installation using the following sequence of windows:

   ```
   Software Installation and Maintenance
     Install and Update Software
       Update Installed Software to Latest Level (Update All)
   ```

   Alternatively, you can use the fastpath command to select the appropriate device:

   ```
   smitty update_latest
   ```

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

4. Select **/dev/cd0 (CD-ROM Drive)**.

5. Select **OK** to display the parameters for **Update All**.

6. Update all previously installed software for MQSeries by selecting the **_update_all** option in the **Software to update** field.

7. Press Enter.

8. Select **OK** in the confirmation window to start updating the software.

Once all previously installed filesets have been updated to the latest level, you can install any additional filesets. See "Custom installation" on page 34 for more information.

Changes to the installation path

Changes in AIX LPP Version 4 packaging mean that the MQSeries for AIX V5.1 client installs into directory **/usr/mqm**, whereas previous versions of the product installed into directory **/usr/lpp/mqm**.

Installation of the MQSeries for AIX V5.1 client will fail if a file system mounted as **/usr/lpp/mqm** is detected. If you are migrating from an earlier version and a file system exists for this directory, you will need to do one of the following things before installing the MQSeries for AIX V5.1 client. Either:

- Uninstall your existing MQSeries client, and either delete the file system or move it to the new install path of **/usr/mqm**

or

- Move the old file system of **/usr/lpp/mqm** to the new installation path of **/usr/mqm** and create a symbolic link from the old path to the new by issuing the following command:

   ```
   ln -s /usr/mqm /usr/lpp/mqm
   ```

If you uninstall your existing client and either delete or move your existing file system, you can then install the MQSeries for AIX V5.1 client as described in "Custom installation" on page 34 or "Easy installation" on page 33.

However, if you move the old file system to the new installation path, you should then perform the migration installation described in "Migrating from an earlier version of MQSeries for AIX" on page 34.

Note: If you have already symbolically linked a file system to **/usr/lpp/mqm**, installation of the MQSeries for AIX V5.1 client will destroy the contents of the file system and the symbolic link, leaving an empty file system. If this happens, you are advised to uninstall your existing MQSeries client and either delete the file system or relink it to the new install path of **/usr/mqm**, before installing the MQSeries for AIX V5.1 client.

The installation process for the MQSeries for AIX V5.1 client creates a symbolic link from the old install path (**/usr/lpp/mqm**) to the new install path (**/usr/mqm**). Therefore any existing scripts or makefiles that reference the old path are still valid.

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.

First, check the initial locale setting for your machine by typing:

```
smitty mle_cc_cust_hdr
```

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

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 install the message catalog for another national language:

1. Type `smit`
2. Follow this sequence of windows:

   ```
   Software Installation and Maintenance
     Install and Update Software
       Install and Update from ALL Available Software
   ```
3. Press the **List** button to display the **Single Select List** window.
4. Select:

   /dev/cd0 (CD-ROM Drive)
5. Press the **List** button on the **Software to Install** field.
6. Select the message catalog that you want to install.
7. Press **OK** to install the chosen message catalog or catalogs.

Translated messages

Messages in U.S. English are always available. If you require one of the other languages that is supported by MQSeries for AIX, 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
export NLSPATH=/usr/lib/nls/msg/%L/%N
```

Removing an MQSeries client from AIX

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

1. Type smit

2. Follow this sequence of windows:

   ```
   Software Installation and Maintenance
     Software Maintenance and Utilities
       Remove Installed Software
   ```

3. Press the **List** button to display the installed software and select the filesets to remove. The MQSeries client filesets have titles starting mqm*.

Installing on Compaq Tru64 UNIX

If you plan to install an MQSeries client and server on the same machine, you should use the appropriate MQSeries Server CD-ROM, see "Installing MQSeries clients on the same machine as the server" on page 30.

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

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

Before installation

Before you can install an MQSeries client on your Compaq Tru64 UNIX system you:
- Must create a group with the name mqm.
- Must create a user ID with the name mqm.
- Are recommended to create and mount a /var/mqm file system, or /var/mqm and /var/mqm/errors file systems.

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

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

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

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

Installation

Carry out the following procedure:
1. Mount the appropriate CD-ROM by typing the following commands:
   a. For Compaq Tru64 UNIX Version 4.0:
      ```
      mount -r -t cdfs -o noversion,rrip /dev/rz18c /cdrom
      ```
      substituting the name of your CD-ROM device for *rz18c*.
   b. For Compaq Tru64 UNIX Version 5.0:
      ```
      mount -r -t cdfs -o noversion,rrip /dev/disk/cdrom0C /cdrom
      ```
      substituting the number of your CD-ROM device for *0C*.
2. Use the Compaq Tru64 UNIX program setld to install the software by carrying out the following procedure:
   a. Type setld -l /cdrom
   b. Press the Enter key.

c. You are prompted for a list of components to install. Select the ones you require, including MQSeries Client. If you want to install all the components, select **all**.

The components you can select are:

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

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

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

Translated messages

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

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

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

Uninstalling an MQSeries client from Compaq Tru64 UNIX

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

```
setld -i
```

and

```
setld -d
```

List the MQSeries components using the Compaq Tru64 UNIX command:

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

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

```
MQS_BASE         installed   IBM MQSeries Base subset
MQS_CLIENT       installed   IBM MQSeries Client subset
MQS_IGRUNTIME    installed   IBM MQSeries Internet Gateway runtime subset
MQS_IGRUNTIMED   installed   IBM MQSeries Internet Gateway runtime data
MQS_IGSAMPLE     installed   IBM MQSeries Internet Gateway samples subset
MQS_IGSAMPLED    installed   IBM MQSeries Internet Gateway sample data
MQS_JAVABASE     installed   IBM MQSeries Java Base subset
MQS_JAVABINDING  installed   IBM MQSeries Java Bindings subset
MQS_JAVACLIENT   installed   IBM MQSeries Java Client subset
MQS_LANG_DE_DE   installed   IBM MQSeries Language-German catalog
MQS_LANG_ES_ES   installed   IBM MQSeries Language-Spanish catalog
MQS_LANG_FR_FR   installed   IBM MQSeries Language-French catalog
MQS_LANG_IT_IT   installed   IBM MQSeries Language-Italian catalog
MQS_LANG_JA_JP   installed   IBM MQSeries Language-Japanese catalog
```

```
MQS_LANG_KO_KR   installed   IBM MQSeries Language-Korean catalog
MQS_LANG_PT_BR   installed   IBM MQSeries Language-Brazilian Portuguese catalog
MQS_LANG_ZH_CN   installed   IBM MQSeries Language-Simplified Chinese catalog
MQS_LANG_ZH_TW   installed   IBM MQSeries Language-Traditional Chinese catalog
MQS_MAN          installed   IBM MQSeries Manual pages
MQS_RUNTIME      installed   IBM MQSeries Runtime subset
MQS_SAMPLES      installed   IBM MQSeries Sample subset
MQS_SERVER       installed   IBM MQSeries Server subset
```

Issue the following command to remove a component:

```
setld -d component name
```

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

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

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.

Components for DOS

The components you can install on DOS 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

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 subdirectories containing the DOS client.
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 6. Verifying the installation" on page 105.

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 30.

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

Components for HP-UX

The components you can install on HP-UX 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 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 the *MQSeries Using Java* book.

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

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

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

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`

  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 appropriate HP-UX documentation.

If you are using HP-UX V10.x, the depot to use is in the `HPUX10/MQS510.000.V10` file under the mount point. If you are using HP-UX V11.x, the depot to use is in the `HPUX11/MQS510.000.V11` 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 Web site for a SupportPac™ that gives additional performance information - see "MQSeries information available on the Internet" on page 185.

Now go to "Chapter 6. Verifying the installation" on page 105.

Translated messages

Messages in U.S. English are always available. If you require one of the other 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 30. If you currently have a manually installed MQSeries client on your OS/2 Warp system from a previous release of MQSeries, you must delete it manually before attempting to install the Version 5.1 client. Before you delete the previous version, save your MSQ.INI file because the deletion process will delete this file.

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.

Components for OS/2 Warp

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

MQSeries Client
The MQSeries client code for your platform.

MQSeries Development 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 Java 1.1.1 (or later compatible version) runtime code on your machine. 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 185.

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

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

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

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 Warp window (or start a full-screen session).

Installing on OS/2 Warp

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.

   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. 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 might 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. This 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. 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 Warp 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).

15. To obtain MQSeries messages in the correct language two environment variables must be set correctly. The installation process sets the environment variable NLSPATH including <installation directory>\LOCALE\%L\%N, where <installation directory> is the directory in which you choose to install the product. You must set the environment variable LANG=<lang>, where <lang> is the subdirectory in which the message catalog for your language is installed, selecting a language from the list below:

Directory	Language
En_US	U.S. English
Pt_BR	Brazilian Portuguese
Fr_FR	French
De_DE	German
It_IT	Italian
Ja_JP	Japanese
Ko_KR	Korean
Es_ES	Spanish
Zh_CN	Simplified Chinese
Zh_TW	Traditional Chinese

Now go to "Chapter 6. Verifying the installation" on page 105.

Unattended installation on OS/2 Warp

You can install MQSeries clients on OS/2 Warp workstations without user interaction being required at the workstation. You can perform this unattended

installation using response files. The sample response file AMQISMC2.RSP shows an example of unattended client installation.

This kind of automatic installation is particularly useful for installing clients over a network because installation can be performed from a redirected drive on a LAN server.

> **Attention**
>
> Note that the INSTALL.EXE file in the root directory of the CD-ROM *must not* be used to perform unattended installations.
>
> You *must* instead invoke the INSTALL.EXE file located in the directory corresponding to the language version of the product that you want to install.

Starting unattended installation

The following steps describe how to perform an unattended installation:

1. Connect to the drive containing the MQSeries product software. This can be a local CD-ROM drive containing the MQSeries client CD, or a remote network drive. For example, F:.

2. Change to the directory appropriate to your language. The directories are as follows:

Directory	Language
\OS2\En_Us	U.S. English
\OS2\Pt_BR	Brazilian Portuguese
\OS2\Fr_FR	French
\OS2\De_DE	German
\OS2\It_IT	Italian
\OS2\Ja_JP	Japanese
\OS2\Ko_KR	Korean
\OS2\Es_ES	Spanish
\OS2\Zh_CN	Simplified Chinese
\OS2\Zh_TW	Traditional Chinese

3. Change to the subdirectory `samples`.

4. Edit the supplied response file, AMQISMC2.RSP for client installation, or create a new response file. The supplied response file can be used for install and delete actions. For more information on response files, see "Installation response files" on page 50

5. Type `install` together with the required parameters (described in "Installation and maintenance parameters" on page 49). For example:

```
INSTALL /A:I /R:J:\INSTMQS\AMQISMC2.RSP
        /S:J:\INSTMQS\en_us
        /L1:J:\INSTMQS\MQM.OUT
        /L2:J:\INSTMQS\MQM.HIS /X
```

This example uses U.S. English. To use another language, substitute `en_us` with the directory appropriate to your language. When you issue this command, you must type it as one continuous line. In the example above, for clarity, the command is spread over several lines.

In this example, the MQSeries client is installed from the redirected drive J: on the server, according to the options supplied in the response file AMQISMC2.RSP (See

"Installation response files" on page 50 for more details). Note that drive J: must be a writeable drive that you must create, and the .RSP file must be put on that drive. The installation log files will also be created on the J: drive. The response file specifies the drive and directory in which to install the client. Errors are logged in MQM.OUT and the history log is contained in MQM.HIS. The /X parameter specifies that the installation is non-interactive.

The example also shows that you must have the INSTALL.EXE available to your system. This file is supplied on the client CD-ROM, but is not installed onto your system. You must make a copy of this file available when you perform maintenance updates on your system.

Installation and maintenance parameters

The installation and maintenance parameters are as follows:

```
INSTALL /A:action
        /G:include path
        /L1:error log
        /L2:history log
        /R:response file
        /S:source location
        /T:install target directory
        /TU:update CONFIG.SYS directory
        /X
```

Note: You can enter the parameters in any order. Equals signs (=) can be used instead of colons (:) in the parameters. Values can be upper or lower case.

/A:`action`
   Specifies the action to be performed by the installation program. If you specify this parameter, the main window of the installation program is not displayed. Valid values for `action` are:
   D   Delete an installed MQSeries for OS/2 Warp system
   I   Install a new MQSeries for OS/2 Warp system
   R   Restore a backed up MQSeries for OS/2 Warp system
   U   Update an installed MQSeries for OS/2 Warp system

   Note: The installation program is not supplied with a corrective service medium. Therefore, if you are updating, restoring, or deleting an MQSeries for OS/2 Warp system, you must have access to the installation program used to install it.

/G:`include path`
   Specifies the drive and path of a general response file to be included by the specific response file. For more information about response files, see "Installation response files" on page 50.

/L1:`error log`
   Specifies the drive, path, and file name of the error log file. The error log contains messages associated with installation, including confirmations and error messages. Messages are written to the error log if you specify the /X parameter.

   You should specify the drive and path where the installation program is running. If you do not specify the /L1 parameter, no error log is maintained. If the error log already exists, it is appended to.

   Example:

```
/L1:D:\LOG\INSMQM.OUT
```

/L2:*history log*
> Specifies the drive, path, and file name of the history log file. The history log contains an entry for each file transferred, each object created, and each installation exit run.
>
> You should specify the drive and path where the install program is running. If you do not specify the /L2 parameter, no history log is maintained. If the history log already exists, it is appended to.
>
> Example:
> ```
> /L2:D:\LOG\INSMQM.HIS
> ```

/R:*response file*
> Specifies the drive, path, and file name of a response file; see "Installation response files".
>
> Example:
> ```
> /R:L:\MQMINS\AMQISAM2.RSP
> ```

/S:*source location*
> Specifies the drive and path containing the source files to be installed or updated.

/T:*install target directory*
> Specifies the drive and path that MQSeries for OS/2 Warp files are installed on to. If you specify this parameter, it overrides the FILE path, which must be specified in the response files.

/TU:*update CONFIG.SYS directory*
> Specifies the drive and path of the target CONFIG.SYS to be updated.

/X Specifies that the installation is fully automatic.

> When you specify this parameter, no progress indicator panel is shown and error messages are logged in the error log file. (You specify the path name of the error log file using the /L1 parameter.) If you do not specify all of the information required for the action to complete, an error occurs.
>
> If you do not specify the /X parameter, the user is prompted for any information that the install program needs to complete the action. In this interactive mode of operation, progress indication is shown and error messages are displayed in secondary windows.

Installation response files

An installation response file is an ASCII text file containing answers to the options that you select when you install or maintain an MQSeries for OS/2 Warp system. This allows installation and maintenance to be performed automatically, without interaction.

In an installation response file you can specify:
- Whether CONFIG.SYS should be updated automatically
- The MQSeries for OS/2 Warp components to be installed
- The path for installation or maintenance
- Whether existing files should be overwritten
- Whether only backup versions of MQSeries for OS/2 Warp should be deleted

The installation response file supplied with MQSeries for OS/2 Warp can be found as a sample file on the installation media. You can use this file to install or delete MQSeries for OS/2 Warp. To perform other actions you must prepare your own installation response files using a suitable editor.

Normally you have only one response file, specified by the /R parameter of the installation program. However, if you need to install or update MQSeries for OS/2 Warp on a workstation with different options, you can use two files. One would be a specific response file that contains options specific to a particular workstation, and the other a general installation response file that contains options common to all workstations.

For example, you might use a general and a specific response file to install a particular component only on some workstations.

You use the /R parameter to specify the specific installation response file, and the /G parameter to specify the location of the general response file.

Response file structure

There are two kinds of line in a response file:

- **Comment lines**

  Comment lines are either blank or start with an asterisk (*) or a semicolon (;).

- **Response lines**

  Response lines are used to determine the options and configurations to install on the target system. Response lines have the following syntax:

  keyword = value

  Keyword-value pairs can be in any order. However, there can be only one pair per line. You can enter keywords in upper or lower case letters. You cannot include spaces within keywords.

The maximum line length in a response file is 255 characters.

Keywords for response files

The following keywords are supported in response files for MQSeries for OS/2 Warp:

CFGUPDATE

Specifies whether CONFIG.SYS is updated automatically. Valid values for this keyword are:

> **AUTO**Automatically updates CONFIG.SYS
> **MANUAL**Does not update CONFIG.SYS

Actions: INSTALL, UPDATE, DELETE.

COMP

Specifies the name of a component on which to perform an action. See the server or client component lists for valid values.

You do not need to enclose the product names within quotes.

Actions: ALL actions.

DELETEBACKUP

Specifies whether to delete only the backup version of MQSeries for OS/2 Warp or the entire product. Valid values are YES and NO.

When you remove MQSeries for OS/2 Warp, none of your user-supplied information, for example queue manager data, is deleted. This is particularly important if you intend to delete and reinstall the product, because your previous queue manager data will remain.

This can lead to an unexpected directory and file structure, if you attempt to recreate queue managers with identical names to those used in the previous installation, because such data might already be present.

Actions: DELETE.

FILE Specifies the drive and directory for MQSeries for OS/2 Warp code. For example the C:\MQM directory.

Actions: INSTALL.

WORK

Specifies the drive and directory for MQSeries data files. For example the C:\MQM directory.

Actions: INSTALL.

INCLUDE

Specifies which general response files to include with a specific response file. The format of this keyword is:

```
INCLUDE = filespec
```

Where *filespec* specifies the general response file to be included. If the file specification contains any global characters (* or ?), the first file found that matches the specification is included. If the specification is not valid, no general response file is included.

Note: You should not have more than five levels of included response files.

Actions: All actions.

OVERWRITE

Specifies whether to overwrite files automatically during installation. Valid values for this keyword are YES and NO.

Actions: INSTALL, UPDATE.

SAVEBACKUP

Specifies whether to save a backup version of MQSeries for OS/2 Warp when it is updated. Valid values for this keyword are YES and NO.

Actions: UPDATE.

Removing an MQSeries client from OS/2 Warp

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 30.

Components for Sun Solaris

The components you can install on 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 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 the *MQSeries Using Java* book.

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

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

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

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`

  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

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

Carry out the following procedure:

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

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

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

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

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

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

   a. Type `pkgadd -d /cdrom/mqclient/solaris/mqs510.img`.

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

      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. You also require Version 2.6 or later of the Sun Solaris operating system.

   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 Web site for a SupportPac that gives additional performance information - see "MQSeries information available on the Internet" on page 185.

Now go to "Chapter 6. Verifying the installation" on page 105.

Translated messages

Messages in U.S. English are always available. If you require one of the other languages 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 must remove the product using the **pkgrm** program.

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

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

Installing on Sun Solaris, Intel Platform Edition

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

Before Installation

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

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

  If you create separate partitions, the following directories *must* be on a local file system:
  - /var/mqm
  - /var/mqm/log

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

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

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

Installation

Carry out the following procedure:

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

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

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

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

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

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

   a. Type pkgadd -d /cdrom/mqclient/solaris/mqs510.img.

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

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

   c. Press the Enter key.

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

Uninstalling the MQSeries for Sun Solaris, Intel client

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

Kernel Configuration

See the MQSeries family Web site for a SupportPac™ that gives additional performance information - see "MQSeries information available on the Internet" on page 185.

Translated Messages

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

For example:

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

Installing on Windows 3.1

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

Components for Windows 3.1

The components you can install on 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

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
   setuppt - Brazilian Portuguese
   setupfr - French
   setupde - German
   setupit - Italian
   setupjp - Japanese
   setupko - Korean
   setupes - Spanish
   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.

   To select a component, click in the box 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 95 or Windows 98

To install an MQSeries client on a Windows 95 or a Windows 98 system you use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

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 30.

Note: The client installation will fail if the PATH variable that has been set up is greater than 267 characters long.

Components for Windows 95 or 98

The components you can install on Windows 95 or 98 systems are:

MQSeries Client
> The MQSeries client code for Windows 95 and Windows 98.

MQSeries Toolkit
> This includes:
> - Sample programs
> - Header files that you can use when writing applications to run on the client

Installation

1. Insert the MQSeries Client CD-ROM into the CD-ROM drive.
2. The installation automatically starts 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.

   To select a component, click in the box 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 MQSeries client for Windows 95 and Windows 98. 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".

Now go to "Chapter 6. Verifying the installation" on page 105.

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 95 and Windows 98" on page 63.

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.

Unattended installation on Windows 95 and Windows 98

You can install a Windows client on a remote machine without interaction. This process is known as *unattended installation*, sometimes referred to as *silent installation*, and uses response files.

An installation response file is an ASCII text file containing values for the options that you select when you install an MQSeries for Windows system.

For unattended installation, you must:

1. Carry out an installation on a machine and record the actions performed to install the product.

   In this example, the machine is being used as a server – see "Setting up the server and creating a response file".

2. Use the recording you have created to install the product on a remote machine. See "Installing on the remote machine" on page 62.

Setting up the server and creating a response file

To set up your server and create a response file, follow these steps:

Installing on Windows 95 or 98

1. Load the MQSeries client CD-ROM. When the *Language Selection* window appears, select **Cancel**.
2. Copy the *win95* or *win98* subdirectory, and all of its subdirectories, from the client CD-ROM to a shared-resource image directory.
3. Select one of the following language directories:
   - SetupCn – Simplified Chinese
   - SetupDe – German
   - SetupEn – English
   - SetupEs – Spanish
   - SetupFr – French
   - SetupIt – Italian
   - SetupJp – Japanese
   - SetupKo – Korean
   - SetupPt – Brazilian Portuguese
   - SetupTw – Traditional Chinese
4. From the selected language directory, run `setup.exe`, with the `-r` option to create a response file.

   This can be done by:
   a. Highlighting `setup.exe`
   b. Dragging it to the **Start**, **Run** dialog
   c. Adding **-r** to the end of the fully-qualified file name
   d. Selecting **OK** to start the process
   e. Selecting the components that you want to install

   Notes:
   a. When the installation has ended, a response file will have been automatically generated that can be used as the basis for unattended installation.
   b. The response file SETUP.ISS will be generated in the WINDOWS directory for Windows 95 or Windows 98.
5. When the response file has been generated, you can copy it to the language dependent directory in the shared-resource image directory for your national language.

   For example, to copy the SETUP.ISS file from the WINNT35 directory to \IMAGE\SETUPEN, type:

   ```
   COPY C:\WINNT35\SETUP.ISS M:\INSTMQS\IMAGE\SETUPEN
   ```

   where SETUPEN is the appropriate language directory for your system.

   This assumes that the Windows 95 or 98 operating system was installed to the C drive and that the MQSeries installation image on the server resides on M:\INSTMQS.

Now you have set up your server, you can go on to install on the remote machine.

Installing on the remote machine

Unattended installation is particularly useful for installing a client over a network because you can do it from a redirected drive on a LAN server.

The remote machine *must* have access to the shared-resource image directory, which includes the:
- Appropriate language subdirectory
- Disk subdirectories

To connect the workstation to the redirected drive on the server, follow these steps:

1. Use the NET USE command as follows:

   NET USE *devicename* *servername**netname*

   For example:

   > NET USE J: \\MQMNT\MQMSHARE

   where:
   - J: is the logical drive name to be connected to the shared resource directory.
   - MQMSHARE is the netname for M:\INSTMQS.
2. Go to the language directory that contains the SETUP.EXE and SETUP.ISS files and run **SETUP -s**.

Notes:

1. The -s parameter indicates that the installation is to be carried out in silent mode. This parameter is mandatory for an unattended installation.
2. SETUP.EXE uses the SETUP.ISS file in the current directory to silently install the product.

Now go to "Chapter 6. Verifying the installation" on page 105.

Removing an MQSeries client from Windows 95 and Windows 98

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.

Installing on Windows NT

See the following sections for information on how to install an MQSeries Client on Windows NT:

- "Installating MQSeries Clients on a separate machine from the server"
- "Installing MQSeries clients on the same machine as the server" on page 30
- "Installing from a LAN" on page 66
- "Using the System Management Server with MQSeries for Windows NT" on page 66
- "Unattended installation on Windows NT" on page 68

To install an MQSeries client on Windows NT, use the MQSeries Client CD-ROM supplied as part of the MQSeries product.

Components for Windows NT

The components you can install on Windows NT systems are:

MQSeries Client
    The MQSeries client code for your platform.

MQSeries Development 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 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 Warp 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 185.

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

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

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

Installating MQSeries Clients on a separate machine from the server

To install an MQSeries client for Windows NT, you must be logged on to Windows NT as an administrator.

MQSeries checks for any existing MQSeries configuration files (MQS.INI). If it finds any, it automatically migrates configuration information to the Windows NT registry. Otherwise, MQSeries automatically puts its configuration information directly into the Windows NT registry.

Installation procedure

1. Insert the MQSeries Client CD-ROM into the CD-ROM drive.

2. The installation automatically starts 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 **OK** 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. Choose Installation Folders lets you choose folders into which the MQSeries program files and data 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. Select Components displays a list of components from which you can select the ones you want to be installed.

   The component **Java Client** should be installed only if you have Java 1.1.1 (or later compatible) runtime code on your machine.

   To select a component, click in the box 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 on the panel, and the space available on the drive you have selected.

   If you deselect a previously installed component it is removed. 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 IBM MQSeries Client. You can rename the default or choose an existing folder.

8. Ready to Copy 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 Setup Complete window appears when the selected components have been installed. Click the Finish button or press Enter to close the window (after optionally selecting Launch Notepad to View the Release Notes).

10. The installation of the MQSeries client is now complete.

11. When setup is complete, the IBM MQSeries folder is added to the Start menu. Note that MQSeries Clients are sets of services and do not have to be explicitly run, so the folder does not have an object called "client".

Now go to "Chapter 6. Verifying the installation" on page 105.

Running setup again

You can run the installation again to add another component, refresh a component you have already installed, or to remove a component.

Installing from a LAN

There are two ways to put MQSeries installation files on a LAN file server for easier access: you can make the MQSeries for Windows NT Client CD-ROM drive shareable, or you can copy the installation files from the CD-ROM to a file server, by following these steps:

1. Create a folder on the LAN file server to store the installation files. For example:

   ```
   md m:\instmqs
   ```

2. Load the MQSeries for Windows NT client CD-ROM. If you have autorun enabled, the language selection panel will appear. Cancel this panel.

3. Copy the Winnt directory structure from the CD-ROM to the m:\instmqs folder. For example:

   ```
   xcopy e:\Winnt\*.* m:\instmqs /e
   ```

   You can save space on the hard drive by copying only the subfolders for the languages that you require. The language subfolders are:

SetupCn	Simplified Chinese
SetupDe	German
SetupEn	U.S. English
SetupEs	Spanish
SetupFr	French
SetupIt	Italian
SetupJp	Japanese
SetupKo	Korean
SetupPt	Brazilian Portuguese
SetupTw	Traditional Chinese

4. Give all licensed users access to the folder that now contains the CD-ROM image (in this example, the m: drive).

5. From a command prompt on the target machine, connect to the appropriate drive and folder using the `net use` command:

   ```
   net use devicename \\servername\netname
   ```

   For example:

   ```
   net use x: \\mqmnt\instmqs
   ```

   where x: is the required mapped drive on the target machine. Alternatively, use Windows NT Explorer or another method to map the shared resource to a drive letter.

6. Change to the installation directory (in this example x), type `Setup` and press Enter.

7. Follow the prompts.

Using the System Management Server with MQSeries for Windows NT

This section describes how to install, or remove, an MQSeries for Windows NT client using the System Management Server (SMS).

Creating SMS packages and jobs for MQSeries

You must create:

- An SMS software package containing the MQSeries software.
- An SMS job to distribute and install the software package (see "Creating the MQSeries SMS job").

For more detailed information on how to create a software package and a job, refer to the Microsoft System Management Server documentation.

Creating the MQSeries SMS software package

To create the SMS software installation package:

1. From the Microsoft SMS Administrator application, open the **Packages** folder and then create a new package.
2. In the SMS **Package Properties** dialog click on the **Import** button to create the software package by importing a Package Definition File (PDF).
3. In the **File Browser** dialog, select the drive where the IBM MQSeries client CD-ROM is located.
4. Select the Winnt directory, which contains the package definition file MQSERIES.PDF.

   You can also find the MQSERIES.PDF file in the local drive, or shared network drive where you copied the MQSeries Installation software.
5. Select the **MQSERIES.PDF** file and click on the **OK** button.
6. Click on the **Workstation** button. In the **Source Directory** entry field, specify the fully-qualified path name to the MQSeries root directory that contains the MQSeries installation software.
7. Select the appropriate Workstation Command Line:

   - **Automated Uninstallation of IBM MQSeries - Windows NT client**
   - **Automated Installation of IBM MQSeries - Windows NT client (US English)**
8. Click on the **Properties** button for each process and review the **Command Line** entry field to ensure that the parameters are correct.

   Note: The -i<miffilepath> parameter specifies the full path and file name of a Management Information Format (.mif) file that installation and uninstallation can generate. Remove the parameter if you do not want to generate a .mif file.
9. Click on the **Close** button to close the **Workstation Properties** dialog.

   Note: If you specified a local path in the **Source Directory** entry field, you get a pop-up dialog warning you that the local path you specified might not be accessible to SMS components running on another machine. Click on the **OK** button to continue.
10. Click on the **OK** button to close the **Package Properties** window.

    A pop-up dialog appears indicating that SMS will update the software package at all sites. Click the **OK** button to continue.

The software package has been created and can be installed by creating an SMS job.

Creating the MQSeries SMS job

You must now create an SMS job to distribute and install the software packages you created that contain the MQSeries installation software.

Installing on Windows NT

Refer to the Microsoft System Management Server documentation for detailed information on how to create and run a job.

Notes:

1. You *must* be logged onto the target machine with Administrator authority in order to install the MQSeries Server.
2. When creating an SMS Job to distribute and install the software package, ensure that you select the appropriate workstation command. The workstation commands are displayed on the **Job Details** dialog in the **Run Phase** section and appear in a drop-down list box.

Unattended installation on Windows NT

You can install MQSeries for Windows NT on a remote machine without interaction, provided that the remote machine can share the client CD-ROM, or a copy of the files on it, and that you can execute a command on the remote machine. This process is called unattended (or silent) installation, and is particularly useful for installing MQSeries for Windows NT clients over a network because you can do it from a shared drive on a LAN file server.

Because there is no user interaction, unattended installation uses a response file. A response file is an ASCII text file containing values for the installation options you want to select. For information on the format of response files, see "Appendix A. Installation response file format for Windows NT" on page 167.

There are three ways to generate a response file:

1. Edit the response file (setup.iss) supplied in the Winnt directory on the MQSeries for Windows NT client CD-ROM, using an ASCII file editor.
2. Generate your own response file using an ASCII file editor.
3. Carry out an installation on a machine and record the options selected to install the product in a response file. To do this you must run setup.exe with the -r parameter, and optionally, the -noinst parameter:
   a. Load the MQSeries for Windows NT client CD-ROM. If you have autorun enabled, the language selection panel will appear. Cancel this panel.
   b. Run setup.exe from the root folder of the CD-ROM, with the -r parameter. This can be done by dragging setup.exe from Windows NT Explorer to the Run dialog in the start menu and adding -r to the end of the file name, or by typing the fully-qualified path to setup.exe into the Run dialog, followed by -r.
   c. The language selection dialog reappears. Perform an installation as you want it performed on the remote machine.
   d. If you are asked to restart the system, you do not need to do so if you are only generating a response file.

   The -noinst parameter (which, if present, must precede the -r parameter) suppresses the MQSeries installation; Setup generates only a response file. The installation creates a response file called setup.iss in the Windows directory, normally c:\winnt. Save this file; you can edit it if necessary.

Use the response file you have created to install the product on a remote machine. (See "Installing on the remote machine".)

Installing on the remote machine
The remote machine must have access to a shared resource or drive on a file server that contains the client CD-ROM or a copy of it. Perform the procedure described

in "Installing from a LAN" on page 66 up to the point where you run Setup. Copy your response file to a location on the file server that is accessible from the remote machine. You can now start the installation on the remote machine:

1. On the remote machine, go to the Setupxx folder on the shared resource.
2. Substitute Setupxx with the name of the language subfolder for the language that you require.
3. Run setup.exe:

   ```
   setup -g<logfile> -f1<responsefile> -f2<secondarylogfile> -s
   ```

   This installs in the language specified by Setupxx.

In the above command:

<logfile>
> The full path to an installation log file. Setup creates a U.S. English ASCII text log file containing details of what happens during installation. You should check this file to see if any errors occurred. If you omit the -g<logfile> parameter, Setup creates a file called amqilogn.txt in the data-files folder on the machine running the installation. In this case, any messages generated before the data-files folder is created are lost. The -g<logfile> parameter must be placed before the -f1 and the -s parameters, otherwise it is ignored.

<responsefile>
> The full path to the response file you prepared. If you omit the -f1<responsefile> parameter, the response file must be in the Setupxx language subdirectory.

<secondarylogfile>
> The full path to a secondary installation log file. This file contains less detail than the other log file and should not be used as the primary source for information about the success of the installation. If you omit the -f2<secondarylogfile> parameter, Setup attempts to create a file called setup.log in the language subdirectory (Setupxx), which is not possible if the installation is being performed from a CD-ROM. The -f2<secondarylogfile> parameter must be placed after the -f1 parameter.

-s This parameter tells Setup to run in silent mode.

Enclose the long path name and file name expressions in double quotes.

Now go to "Chapter 6. Verifying the installation" on page 105.

Removing an MQSeries client from Windows NT

Any of the following steps can be used to remove the MQSeries client files from your machine:

- From the Start menu, select Settings/ Control Panel/ Add-Remove
- Run the MQSeries Clients uninstallation programme from Windows or from the command prompt
- Use SMS
- Run unattended installation

If you want to remove some or all of the MQSeries client files from your machine, select IBM MQSeries Client, which launches the install program. Alternatively,

Installing on Windows NT

choose MQSeries Uninstallation from the IBM MQSeries client folder. You can choose to uninstall one or more components or the whole of the MQSeries client, including or excluding data.

For more information on uninstalling an MQSeries for Windows NT client, see "Appendix B. Uninstalling on Windows NT" on page 171.

Chapter 4. Installing MQSeries clients with non-Version 5 products, and OS/390

This chapter describes how to install the client components for the following non-Version 5 MQSeries products:
- MQSeries for AT&T GIS UNIX Version 2 Release 2
- MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS Version 2 Release 2
- MQSeries for MVS/ESA Version 1 Release 2
- MQSeries for SINIX and DC/OSx Version 2 Release 2
- MQSeries for Tandem NonStop Kernel, V2.2.0.1
- MQSeries for VSE/ESA Version 2 Release 1

This chapter also describes how to install the client components for MQSeries for:
- MQSeries for OS/390

- MQSeries for MVS/ESA

If you are using an MQSeries Version 5.1 product, see "Chapter 3. Installing MQSeries client components from Version 5 products (not OS/390)" on page 29.

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 "Obtaining MQSeries clients from IBM Transaction Processing SupportPacs" on page 72 for more information about this.

MQSeries for OS/390, MQSeries for Tandem NSK, and MQSeries for VSE/ESA can accept connections from MQSeries clients on other platforms. An MQSeries client cannot run on OS/390, Tandem NSK, or VSE/ESA, 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 "Obtaining MQSeries clients from IBM Transaction Processing SupportPacs" on page 72).

If you are installing the MQSeries client for VM/ESA, you do not need the installation procedure detailed for the other platforms. See "MQSeries client for VM/ESA" on page 80.

This chapter discusses the following topics:
- "Obtaining MQSeries clients from the MQSeries products" on page 72
- "Obtaining MQSeries clients from IBM Transaction Processing SupportPacs" on page 72
- "Installing the MQSeries server" on page 73
- "Installing MQSeries clients from MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS" on page 74
- "Installing MQSeries clients from MQSeries for UNIX systems" on page 77
- "MQSeries client for VM/ESA" on page 80
- "Changing the config.sys and autoexec.bat files" on page 80

Obtaining 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 information, see the *System Management Guide* for your platform.

Alternatively, you can install the MQSeries client from a LAN server on which you have already installed the base product and server (this is 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.

Information about how to copy the required MQSeries client files to the client machine, including the file names and directories, are given in this chapter.

MQSeries client and server on different platforms

Depending on which server platform you are using, a set of clients might be supplied with the base product and server. Clients for DOS, OS/2 Warp, and Windows 3.1 are supplied with MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS, MQSeries for SINIX and DC/OSx, and MQSeries for AT®&T GIS UNIX. These clients are supplied in addition to the files for the MQSeries client on the same platform as your server.

Install MQSeries on the server machine in the normal way, directly from the media (diskette, CD-ROM, or tape, according to your platform). If your server platform is either MQSeries for Compaq (DIGITAL) OpenVMS or one of the UNIX platforms, you can also install the MQSeries client files that you need for your other platform or platforms at the same time. Then you copy the required MQSeries client files to the build and run environment on the client machine. Information about how to do this, including the file names and directories, are given in this chapter.

Clients cannot be run on OS/390, Tandem NSK, or VSE/ESA, so no MQSeries client files are provided on these platforms (see "MQSeries for OS/390, Tandem NSK, and VSE/ESA" on page 73).

Obtaining 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 185.

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.

Installing the MQSeries server

Install the MQSeries base product and server on the machine you want to use as your MQSeries server. Full instructions are given in the *System Management Guide* for your platform.

For MQSeries for OS/390, install the MQSeries base product and client attachment feature on the machine you want to use as your MQSeries server. Full instructions are given in the *MQSeries for OS/390 Program Directory*.

MQSeries client files on the server

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

MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS, and UNIX platforms

When you install MQSeries on MQSeries for Compaq (DIGITAL) OpenVMS, or a UNIX system the files for DOS, OS/2 Warp, and Windows 3.1 clients are supplied. Files for MQSeries clients on the same platform as the server are also supplied.

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

MQSeries for OS/390, Tandem NSK, and VSE/ESA

You can connect MQSeries clients on other platforms to MQSeries for OS/390, Tandem NSK, and VSE/ESA. MQSeries clients cannot be run on these systems, so the files for MQSeries clients are not supplied with them.

If you have only these MQSeries products, and you want to install MQSeries clients on other platforms, see "Obtaining MQSeries clients from IBM Transaction Processing SupportPacs" on page 72.

If you have another MQSeries product available, you might 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 MQSeries for Compaq (DIGITAL) OpenVMS

When you have included the MQSeries client files in the installation of your MQSeries for Compaq (DIGITAL) OpenVMS server machine, the files are located in the following directories:

MQSeries for Compaq (DIGITAL) OpenVMS files

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

OS/2 Warp 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 MQSeries for Compaq (DIGITAL) OpenVMS from MQSeries for Compaq (DIGITAL) OpenVMS

If possible, do this by installing the MQSeries client directly from the media supplied with your MQSeries product. For instructions, see the *MQSeries for Compaq (DIGITAL) OpenVMS System Management Guide*.

Alternatively, you can use the following method:

1. Copy the following files from the MQSeries for Compaq (DIGITAL) OpenVMS server to the MQSeries for Compaq (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. Create the following directory on the MQSeries for Compaq (DIGITAL) OpenVMS client system:

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

   Copy all the files from the same directory on the MQSeries for Compaq (DIGITAL) OpenVMS server into this directory

Installing an MQSeries client on OS/2 Warp from MQSeries for Compaq (DIGITAL) OpenVMS

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

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

2. Copy the following files into the above directories on the OS/2 Warp system from MQSeries for Compaq (DIGITAL) OpenVMS:

   ```
   <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 6. Verifying the installation" on page 105 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 Warp system (for information about how to make the changes, see "Changing the OS/2 Warp config.sys file" on page 81):

   ```
   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 6. Verifying the installation" on page 105.

Installing an MQSeries client on DOS from MQSeries for Compaq (DIGITAL) OpenVMS

1. Create a suitable directory on the DOS system.
2. Copy the following files into the directory you have created on the DOS system from MQSeries for Compaq (DIGITAL) OpenVMS
   `SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.DOS_CLIENT]`:

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

   See "Chapter 6. Verifying the installation" on page 105 for information about how to use the sample executables `amqsputc.exe` and `amqsgetc.exe` (included when you copy *.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 information about how to make the changes, see "Changing the autoexec.bat file for DOS and Windows 3.1" on page 81).
4. Now go to "Chapter 6. Verifying the installation" on page 105.

Installing an MQSeries client on Windows 3.1 from MQSeries for Compaq (DIGITAL) OpenVMS

1. Create a suitable directory on the Windows 3.1 system.
2. Copy the following files, into the directory you have created on the Windows 3.1 system from MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS
   `SYS$COMMON:[SYSHLP.EXAMPLES.MQSERIES.WIN_CLIENT.BIN]`:

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

   See "Chapter 6. Verifying the installation" on page 105 for information about how to use the sample executables `amqsputw.exe` and `amqsgetw.exe` (included when you copy *.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 81).
4. Now go to "Chapter 6. Verifying the installation" on page 105.

Note: You can also install the Windows 3.1 client code on WIN-OS/2® under OS/2 Warp, Windows NT, Windows 95, and Windows 98.

Installing MQSeries clients from MQSeries for UNIX systems

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

> **/opt/mqm/**xxx_client/bin
> **/opt/mqm/**xxx_client/inc
> **/opt/mqm/**xxx_client/lib
> **/opt/mqm/**xxx_client/msg
> **/opt/mqm/**xxx_client/samp/bin

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

OS/2 Warp files

> **/opt/mqm/**os2_client/bin
> **/opt/mqm/**os2_client/dll
> **/opt/mqm/**os2_client/inc
> **/opt/mqm/**os2_client/lib
> **/opt/mqm/**os2_client/msg
> **/opt/mqm/**os2_client/samp/bin

DOS files

> **/opt/mqm/**dos_client/bin
> **/opt/mqm/**dos_client/lib
> **/opt/mqm/**dos_client/inc
> **/opt/mqm/**dos_client/msg
> **/opt/mqm/**dos_client/samp/bin

Windows 3.1 files

> **/opt/mqm/**win_client/bin
> **/opt/mqm/**win_client/dll
> **/opt/mqm/**win_client/inc
> **/opt/mqm/**win_client/lib
> **/opt/mqm/**win_client/msg
> **/opt/mqm/**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 instructions, see the *System Management Guide* for your platform.

Alternatively, you can use the following method:

1. Create the following directory structure on the UNIX system:

   /opt/mqm/bin
   /opt/mqm/inc
   /opt/mqm/lib
   /opt/mqm/msg

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

   /opt/mqm/bin
     /opt/mqm/xxx_client/bin
     /opt/mqm/xxx_client/samp/bin/amqsputc.exe
     /opt/mqm/xxx_client/samp/bin/amqsgetc.exe

   /opt/mqm/inc
     /opt/mqm/xxx_client/inc

Installing from UNIX systems

> **/opt/mqm**/lib
>   **/opt/mqm**/xxx_client/lib
>
> **/opt/mqm**/msg
>   **/opt/mqm**/xxx_client/msg

> See "Chapter 6. Verifying the installation" on page 105 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`.

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

3. Copy the file amq.cat from **/opt/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:

   a. Symbolically link the C header files from **opt/mqm**/inc into /usr/include

   b. Link the libraries from **opt/mqm**/lib into /usr/lib

   c. Link the programs from **opt/mqm**/bin into /usr/bin

6. Now go to "Chapter 6. Verifying the installation" on page 105.

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

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

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

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

   > <drive>:\mqm\bin
   >   **/opt/mqm**/os2_client/bin
   >
   > <drive>:\mqm\dll
   >   **/opt/mqm**/os2_client/dll
   >
   > <drive>:\mqm\inc
   >   **/opt/mqm**/os2_client/inc
   >
   > <drive>:\mqm\lib
   >   **/opt/mqm**/os2_client/lib
   >
   > <drive>:\mqm\msg
   >   **/opt/mqm**/os2_client/msg

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

See "Chapter 6. Verifying the installation" on page 105 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 Warp system (for information about how to make the changes, see "Changing the OS/2 Warp config.sys file" on page 81):

```
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 6. Verifying the installation" on page 105.

Installing an MQSeries client on DOS from a UNIX system

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

```
/opt/mqm/dos_client/bin
  *.exe

/opt/mqm/dos_client/inc
  *.h

/opt/mqm/dos_client/lib
  *.lib

/opt/mqm/dos_client/msg
  *.msg

/opt/mqm/dos_client/samp/bin/amqsputc.exe
/opt/mqm/dos_client/samp/bin/amqsgetc.exe
```

See "Chapter 6. Verifying the installation" on page 105 for information about how to use the sample executables `amqsputc.exe` and `amqsgetc.exe`. Alternatively, you can build these yourself from the corresponding source files.

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 information about how to make the changes, see "Changing the autoexec.bat file for DOS and Windows 3.1" on page 81).

3. Now go to "Chapter 6. Verifying the installation" on page 105.

Installing an MQSeries client on Windows 3.1 from a UNIX system

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

   /opt/mqm/win_client/bin
     *.exe

   /opt/mqm/win_client/dll
     *.dll

   /opt/mqm/win_client/lib
     *.lib

   /opt/mqm/win_client/msg
     *.msg

   /opt/mqm/win_client/inc/
     *.h
     *.hpp (Version 5.1 products)

   /opt/mqm/win_client/samp/bin/amqsputw.exe
   /opt/mqm/win_client/samp/bin/amqsgetw.exe
   /opt/mqm/win_client/samp/bin/imq*c.exe (Version 5.1 products)

   See "Chapter 6. Verifying the installation" on page 105 for how to use the sample executables `amqsputw.exe` and `amqsgetw.exe`. Alternatively, you can build these yourself from the corresponding source files.

   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 information about how to make the changes, see "Changing the autoexec.bat file for DOS and Windows 3.1" on page 81).

3. Now go to "Chapter 6. Verifying the installation" on page 105.

Note: You can also install the Windows 3.1 client code on WIN-OS/2 under OS/2 Warp, Windows NT, Windows 95, and Windows 98.

MQSeries client for VM/ESA

The MQSeries Client for VM/ESA 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 6. Verifying the installation" on page 105.

Changing the config.sys and autoexec.bat files

On OS/2 Warp, 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 following sections explain how to do this.

Changing the OS/2 Warp 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 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 take effect.

Installing MQSeries clients

Chapter 5. 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*. The channels used to connect MQSeries clients to servers are called MQI 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 8. Using channels" on page 119.

Before you define your MQI channels, you will need to:
1. Decide on the form of communication you are going to use. See "Deciding which communication type to use"
2. Define the connection at each end of the channel:
   - Configure the connection.
   - Record the values of the parameters that you need for the channel definitions.
   - Enable the server to detect incoming network requests from your MQSeries client. This involves starting a *listener*.

This chapter explains how to perform these tasks.

Deciding which communication type to use

There are five types of communication for MQI channels on different platforms:
- DECnet
- LU 6.2
- NetBIOS
- SPX
- TCP

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 might 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.

Communication links

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

Transmission protocol	MQSeries client	MQSeries server
TCP/IP	MQSeries for Compaq (DIGITAL) OpenVMS DOS OS/2 Warp UNIX systems VM/ESA Windows 3.1 Windows 95 and 98 Windows NT	AS/400 MQSeries for Compaq (DIGITAL) OpenVMS OS/390 OS/2 Warp Tandem NSK UNIX systems Windows NT VSE/ESA
LU 6.2	MQSeries for Compaq (DIGITAL) OpenVMS OS/2 Warp UNIX systems VM/ESA Windows NT	AS/400 OS/390 OS/2 Warp Tandem NSK UNIX systems (1) Windows NT
NetBIOS	DOS OS/2 Warp Windows 3.1 Windows 95 and 98 Windows NT	OS/2 Warp Windows NT
SPX	DOS OS/2 Warp Windows 3.1 Windows 95 and 98 Windows NT	OS/2 Warp Windows NT
DECnet	MQSeries for Compaq (DIGITAL) OpenVMS	MQSeries for Compaq (DIGITAL) OpenVMS
Note: 1. Except Sun Solaris, Intel Platform Edition		

Defining a connection is described in the following sections:
- "Defining a TCP/IP connection" on page 85
- "Defining an LU 6.2 connection" on page 91
- "Defining a NetBIOS connection" on page 98
- "Defining an SPX connection" on page 100
- "Defining a DECnet connection" on page 102

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 defaults to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to MQSeries.

2. Initialize TCP/IP, and record the network address, in the form 1.11.1.11, of the server machine.

3. Configure files (or run a command) to specify the port number and to run a listener program (not OS/390). On OS/390, start a channel initiator and a listener.

For more detailed step-by-step examples, see the *MQSeries Intercommunication* manual.

TCP/IP connection limits

On any platform, there is a limit to the number of outstanding connection requests that can be queued at a single TCP/IP port. These backlog values are shown in the following table:

Table 3. Maximum outstanding connection requests queued at a TCP/IP port

Server platform	Maximum connection requests
AIX	100
AS/400	512
AT&T GIS UNIX	5
MQSeries for Compaq (DIGITAL) OpenVMS	5
Compaq Tru64 UNIX	10
HP-UX	20
OS/2 Warp	10
OS/390	255
Sun Solaris	100
SINIX and DC/OSx	5
Tandem NSK	5
VSE/ESA	5
Windows NT Server	100
Windows NT Workstation	5

If the connection limit is reached, the client receives a return code of MQRC_Q_MGR_NOT_AVAILABLE from the **MQCONN** call, and an AMQ9202 error in the client error log (mqm\errors\amqerr0n.log). If the client retries the **MQCONN** request, it might be successful.

To increase the number of connection requests you can make, and avoid error messages being generated by this limitation, you can define more than one port on a single queue manager, or define ports on multiple queue managers.

Defining a TCP/IP connection 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.

You can use the KeepAlive option to make TCP/IP periodically check that the other end of the connection is still open. If it is not, the channel is closed. To use this option, create a queue manager configuration file (QM.INI) and add the following entry to it:

```
TCP:
   KeepAlive=yes
```

Store the QM.INI file in the appropriate directory, depending upon the platform:

- OS/2 Warp, Windows 95, Windows 98, Windows NT:

  ```
  <drive>:\<dir>\mqm
  ```

- DOS, Windows 3.1, WIN-OS/2:

  The root directory of the drive where the client is installed

- UNIX systems:

  ```
  /var/mqm
  ```

Note: On Windows NT this information is in the registry. For more information, access the MQSeries Information Center.

Defining a TCP/IP connection on an OS/2 Warp 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`. 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 might 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 square brackets indicate an optional parameter; this is not required for the default queue manager. If MQSeries for OS/2 Warp 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.

Using the Run Listener (RUNMQLSR) command

To run the listener supplied with MQSeries for OS/2 Warp (which 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
```

Defining a TCP/IP connection 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 might 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
```

Using the Run Listener (RUNMQLSR) command

To run the listener supplied with MQSeries for Windows NT (which starts new MQI channels as threads), use the RUNMQLSR command. For example:

TCP/IP connection

```
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.

For

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
```

Defining a TCP/IP connection 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 might
    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:

    ```
    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
```

Using the Run Listener (RUNMQLSR) command

To run the listener supplied with MQSeries for AIX, HP-UX, and Sun Solaris (which 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
```

Defining a TCP/IP connection 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.

Note: There is a limit to the number of outstanding connection requests that can be queued at a single TCP/IP port. This is given in "TCP/IP connection limits" on page 85.

Defining a TCP/IP connection on an OS/390 server

Set up communications for MQSeries for OS/390 to use TCP/IP channels, as described in the *MQSeries Intercommunication* manual. Then:

1. Start the channel initiator

   ```
   START CHINIT
   ```

2. Start the listener

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

Defining a TCP/IP connection on a MQSeries for Compaq (DIGITAL) OpenVMS server

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

See the *MQSeries Intercommunication* manual for information about how to configure TCP/IP services on MQSeries for Compaq (DIGITAL) OpenVMS.

Note: There is a limit to the number of outstanding connection requests that can be queued at a single TCP/IP port. This is given in "TCP/IP connection limits" on page 85.

Defining a TCP/IP connection on a Tandem NSK server

Set up communications for MQSeries for Tandem NonStop Kernel to use TCP/IP channels, as described in the *MQSeries Intercommunication* manual. 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 manual. For example:

- `runmqlsr -t tcp`
- Start server `mqs-tcplis00` from pathway

By default, the MQSeries for Tandem NonStop Kernel listener program uses port 1414 and TCP/IP process `$ztc0`.

See the *MQSeries for Tandem NonStop Kernel System Management Guide* for information about how to configure TCP/IP services on Tandem NSK.

Defining a TCP/IP connection on a VSE/ESA server

Define a TCP/IP connection as follows:

1. Define the listener port number and number of clients to be supported to the VSE/ESA queue manager. You can do this through the MQSeries for VSE/ESA Global System Definition.
2. Ensure that the code page you use is one that is supported for clients. See the *MQSeries Intercommunication* manual for more information.
3. Restart the MQSeries for VSE/ESA queue manager with the MQSE and MQIT transactions.

When the MQSeries for VSE/ESA queue manager is started, the TCP/IP listener starts automatically.

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 (see "Setting the Transaction Program name") and TpPath.
3. Establish a valid SNA session between the MQSeries client and server machines.

On the server

- If you are using OS/390, start a channel initiator and a listener
- On any other platform, start a listener, or create a listening attachment

LU 6.2 on an OS/2 Warp MQSeries client

The following steps will enable you to configure an LU 6.2 connection on an OS/2 Warp MQSeries client.

Configuring SNA

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

Setting the Transaction Program name

Set the Transaction Program name (TpName or TPNAME) as shown in the following table. This table applies to all platforms.:

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

Server platform	TPNAME
OS/2 Warp	As specified in the OS/2 Warp Run Listener command on the server, or defaulted from the OS/2 Warp 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.
UNIX systems	The same as the corresponding TpName in the side information on the remote queue manager on the server. **Note:** UNIX is case-sensitive
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.
OS/390	The same as the corresponding TpName in the side information on the remote queue manager on the server.

For more detailed step-by-step examples, see the *MQSeries Intercommunication* manual.

Establishing a session

Establish a valid session between the two machines. You can specify the local LU that MQSeries for OS/2 Warp 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 Warp the QM.INI file is located in directory C:\MQM.

LU 6.2 connection

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 Warp 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 Warp 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 information about defining 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 Warp program path and file name  :  c:/mqm/bin/amqcrs6a.exe
Program parameter string               :   -n AMQCRS6A
```

This example 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)
```

This example 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 Table 4 on
page 91:

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 (either 2.11 or 3) listed 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 8. Using channels" on page 119).

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

1. 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 *Sunlink P2P LU 6.2 Programmer's Guide*.

2. Set the Transaction Program name (TpName or TPNAME) as shown in the Table 4 on page 91:

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

4.

   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

   Enter the Partner LU Name at the receiving machine, the TpName and the Mode Name in the CPI-C Side Object. 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 system server

1. 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
   ```

2. 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 can specify a user who is a member of the mqm group.

You might need 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 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 3.

```
                        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 3. 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 can 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.

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.

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 the WRKCLS command, for example, to display existing classes or to create a new class.

More information about 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 NonStop Kernel to use LU 6.2 channels, as described in the *MQSeries Intercommunication* manual and in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Ensure that 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 OS/390 server

Set up communications for MQSeries for OS/390 to use LU 6.2 channels, as described in the *MQSeries Intercommunication* manual. Then:

1. Start the channel initiator

   ```
   START CHINIT
   ```

2. Start the listener

   ```
   START LSTR TRPTYPE(LU62) LUNAME(luname)
   ```

LU 6.2 on a MQSeries for Compaq (DIGITAL) OpenVMS client

On MQSeries for Compaq (DIGITAL) OpenVMS, SNA LU 6.2 supports only PU 2.0. Therefore communication can only be to PU 5.0 host. If you want to communicate with a server on a platform other than OS/390 or Tandem NSK, you must use another protocol. See Table 2 on page 84 for information about the protocols you can use.

1. Configure SNA so that an LU 6.2 conversation can be established between the MQSeries client machine and the server machine. See the *MQSeries Intercommunication* manual for information.

2. 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 OS/390 server.

3. 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.

   If you want to use this method, you need to create a QM.INI file and add the following entry to it:

   ```
   NETBIOS:
     LocalName = Your_env_Name
   ```

   Store the QM.INI file in the appropriate directory, depending on the platform:

   OS/2 Warp, Windows 95, and Windows 98
   ```
   <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.

   Windows NT
   This information is in the registry. For more information, see the MQSeries Information Center.

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 Warp 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
   ```

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 System Administration* manual for information about the options on the RUNMQLSR command.

Note: Both the sending end (MQSeries client) and the receiving end (server) *must* have a local NetBIOS name defined. Ensure that all NetBIOS names used are unique in the network; if they are not, unpredictable results might 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 parameter in the registry. The entry in the registry is:

   ```
   NETBIOS:
      LocalName = Your_env_Name
   ```

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 System Administration* manual for information about the options on the RUNMQLSR command.

Note: Both the sending end (MQSeries client) and the receiving end (server) **must** have a local NetBIOS name defined. Ensure that all NetBIOS names used are unique in the network; if they are not, unpredictable results might occur.

Defining an SPX connection

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

SPX on an MQSeries client (any suitable platform)

There is no action required now other than to ensure 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 Warp or Windows NT)

Determine 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 might 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 are suitable. The actual parameters, and the method of changing them, vary according to your platform and the provider of the SPX communications support. The following sections describe some of these parameters, particularly where they could influence the operation of MQSeries channels and client connections.

SPX parameters on an OS/2 Warp client

Refer to the Novell Client for OS/2 Warp 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 might 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 might 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 timeout values adjust the KeepAlive behavior. 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 through 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 might need to increase this value.

SPX parameters on a DOS or Windows 3.1 client

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 might 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 might 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 might 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 parameters on a Windows NT client

Refer to the Microsoft documentation for full information about the use and setting of the NWLink SPX and IPX parameters. The IPX/SPX parameters 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 parameters on an MQSeries client for Windows 95 and Windows 98

Refer to the Microsoft documentation for full information about 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 following sections explain how to define a DECnet connection on the client and server machines, according to the platforms you are using:

DECnet on an MQSeries client

There is no action required now, other than to ensure 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 (MQSeries for Compaq (DIGITAL) OpenVMS)

Decide on an object number or task name.

Determine 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 that 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
$
```

   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
```

   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 directory for the user name specified for the application.

5. The MQSC configuration for CONNAME, as for DECnet Phase IV, is:

```
NODE(APP_OR_OBJ_NAME)
```

DECnet connection

Chapter 6. Verifying the installation

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

This chapter tells you 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" on page 106
2. "Setting up the MQSeries client" on page 108
3. "Putting a message on the queue" on page 109
4. "Getting the message from the queue" on page 111
5. "Ending verification" on page 112

The installation used for the example

These instructions assume that:

- The full MQSeries product has been installed on a server:
  - The Base Product and the Client Attachment feature on OS/390
  - The full MQSeries for VSE/ESA product on VSE/ESA platforms
  - The *Base Product and Server* on other platforms
- The MQSeries client software has been installed on a client machine, and MQSeries client files have been transferred, 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 is more information about this in "Chapter 5. Configuring communication links" on page 83.

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 73.

What the example shows

The following example shows how to create a queue manager called *queue.manager.1* (not on OS/390), 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 7. Setting up MQSeries client security" on page 115 for details if you are concerned with MQSeries client security issues.

Setting up the server

This section does not apply to AS/400, OS/390, Tandem NSK, or VSE/ESA. For information about setting up the server on these platforms, see the sections that follow.

Note: If you are using MQSeries for Windows NT v 5.1 you can use the MQSeries Explorer as well as the command line to carry out most MQSeries tasks. Details of how to use MQSeries Explorer for the verification example are given in "Setting up the server (Windows NT) using MQSeries Explorer" on page 107.

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 5.1 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 Administration Guide* or *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:

   `All valid MQSC commands processed`

   If there are commands that cannot be processed, you need to check your server installation. See the *System Administration Guide* or *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 117 for information about MCAUSER.

7. Stop MQSC by pressing Ctrl+D (on MQSeries Version 5.0 and above type `end`) and then Enter.

8. Configure the system to start channels

   On OS/2 Warp 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 "Defining a TCP/IP connection on a UNIX system server" on page 88 for details of how to do this.

On MQSeries for Compaq (DIGITAL) OpenVMS

See the *MQSeries for Digital OpenVMS System Management Guide* for information about how to configure TCP/IP services to start channels.

Setting up the server (Windows NT) using MQSeries Explorer

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 connection.

- Create a default queue manager
  1. Open MQSeries Explorer from the Start Menu.
  2. Right-click the folder called Queue Managers, and select New.
  3. In the first entry field, type the queue manager name, *queue.manager.1*.
  4. Make this the default queue manager.
  5. Click Next twice.
  6. Create the server connection.
  7. Click Next.
  8. Enter a port number of 1414 and click finish.
- Create a local queue
  1. Expand the queue manager you have just created and right-click queues.
  2. Select new local queue.
  3. Enter the queue name and click OK.
- Define the server-connection channel
  1. Select Advanced, and right-click Channels.
  2. Select New Server-connection channel.
  3. Enter the channel name, CHANNEL1
- Run the listener

  The listener will have automatically started when the queue manager was set up. You can verify this by opening MQServices, and selecting *queue.manager.1*. The listener will be running.

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 MQMNAME('qmgr')
   ```

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

   ```
   CRTMQMQ QNAME(QUEUE1)  QTYPE(*LCL)
   ```

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

   ```
   CRTMQMCHL CHLNAME(CHANNEL1) CHLTYPE(*SVRCN) TRPTYPE(*TCP)
             MCAUSRID('QMQM')
   ```

   Note: QMQM is the default user ID.

5. Start the listener by entering the following command:

```
STRMQMLSR MQMNAME('qmgr')
```

Setting up the server (OS/390)

Customize your MQSeries for OS/390 installation as described in the *MQSeries for OS/390 System Setup 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 might need the Batch/TSO adapter and the operations and control panels to perform administration for clients.

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 (VSE/ESA)

Verify your VSE/ESA server installation as described in the *MQSeries Installation Verification Test* section of the *MQSeries for VSE/ESA System Management Guide*.

Setting up the MQSeries client

When an MQSeries application is run on the MQSeries client, it requires 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, two methods are used to define the client-connection channel;
- "Defining a client-connection channel, using MQSERVER"
- "Defining a client-connection channel, using MQSeries Explorer" on page 109

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 9. Using MQSeries environment variables" on page 131).

- For DOS, OS/2 Warp, Windows 3.1, Windows 95 and Windows 98, 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 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, or the registry for Windows NT. If no value is specified in the QM.INI file, or the registry for Windows NT, 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.

Defining a client-connection channel, using MQSeries Explorer

You can use MQSeries Explorer to define the client-connection if you are setting up the client and server on the same machine.The following steps will define the client-connection end of the channel.

1. Select the queue manager, *queue.manager.1*
2. Select **advanced** then **client-connection** then **new**

   Select a name for the client-connection, and enter the following as the connection name:

   `server-connection(port)`

   Where:

   `server-address` is the TCP/IP hostname of the server
   `(port)` is the TCP/IP port number the server is listening on

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).

Information on errors at this stage can be found in "Error messages with MQSeries clients" on page 158.

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

1. 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 5.1 products. Then enter the following command:

   `amqsputc QUEUE1 qmgr`

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

2. The following message is displayed:

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

3. Type some message text and then press Enter twice.

4. The following message is displayed:

```
Sample AMQSPUT0 end
```

5. The message is now on the queue.

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*.

1. Enter the following command:

```
AMQSPUT0 QUEUE1 qmgr
```

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

2. The following message is displayed:

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

3. Type some message text and then press Enter twice.

4. The following message is displayed:

```
Sample AMQSPUT0 end
```

5. 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

To run AMQSPUTW from the Windows Program Manager:

1. Select the menu item File/Run...

2. 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.

It is important to always look in the output file to see what has happened, because 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.

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 VM/ESA, or Windows 3.1)

1. 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.
2. The message on the queue is removed from the queue and displayed.

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*.

1. Enter the following command:

   ```
   AMQSGET0 QUEUE1 qmgr
   ```

   Where qmgr is the name of the queue manager on the server.
2. 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

To run AMQSGETW from the Windows Program Manager:
1. Select the menu item File/Run...
2. 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

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

Ending verification

The verification process is now complete.

On OS/390, you can stop the queue manager on the server by issuing the `STOP CHINIT` command followed by the `STOP QMGR` command.

On other platforms except VSE/ESA, you can stop the queue manager on the server by typing:

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

If you want to delete the queue manager on the server (not OS/390 or VSE/ESA) type:

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

For VSE/ESA, see the information about the MQSeries installation verification test in the *MQSeries for VSE/ESA System Management Guide*.

Part 3. System administration

System administration

Chapter 7. 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.

Figure 4. Security in a client-server connection

1. Transport level

   This is the same as for two MQSeries queue managers (server to server) and is described in the *MQSeries Intercommunication* manual.

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 *MQSeries Intercommunication* manual.

   DCE security exits are supplied with MQSeries Version 5.1 products for clients on AIX, HP-UX, OS/2 Warp, Sun Solaris, Windows 95, Windows 98, and Windows NT. These are described in the *MQSeries Intercommunication* manual. If no security exits are used, access to MQSeries objects is determined by the server-connection channel definition. See "Access control" on page 117 for details.

3. Identification passed to a channel security exit

Security

In client to server communication, the channel security exits do not have to operate as a pair. The exit on the MQSeries client side can be omitted. In this case the user ID is placed in the channel descriptor (MQCD) and the security exit can alter it, if required. Some clients also send additional information to assist identification.

- For Windows NT and UNIX clients, the user ID that is passed to the server is the currently logged-on user ID on the client. In addition, the Windows NT client passes the security ID of the currently logged-on user.
- For DOS, OS/2 Warp, Windows 3.1, and Windows 95 and Windows 98 clients, he the user ID is specified via the environment variable MQ_USER_ID. In addition, for these clients only, the environment variable MQ_PASSWORD is transmitted to the server.

The values of the user ID and, if available, the password or security ID, can be used by the server security exit to establish the identity of the MQSeries client.

Environment variables

For MQSeries clients on DOS, OS/2 Warp, Windows 3.1, Windows 95, and Windows 98, if a security exit is not defined, the values of two environment variables MQ_USER_ID and MQ_PASSWORD are transmitted to the server. The values are passed in the channel definition (MQCD) to the server security exit when the exit is invoked. The values can then be used by the exit to establish the identity of the MQSeries client.

On these platforms, set the variables in the environment in which the MQSeries client is going to run. Note that MYUSERID and MYPASSWORD must be in uppercase if the client is going to communicate with an MQSeries server on OS/400®. For example:

```
SET MQ_USER_ID=MYUSERID
SET MQ_PASSWORD=MYPASSWORD
```

The MQ_USER_ID and MQ_PASSWORD environment variables are not supported on Windows NT and the UNIX platforms. On these platforms, identification is established when the currently logged-on user ID of the client is passed automatically to the server.

User IDs

If the MQSeries server and client are both on Windows NT, and if the MQSeries server has access to the domain on which the client user ID is defined, MQSeries supports user IDs of up to 20 characters.

If the MQSeries server is on Windows NT, and the client is on a platform that uses the environment variable for specifying the user ID, the user ID can be in the format user@domain. The MQSeries server then retrieves user account information from the specified NT domain. In this case, the maximum length for the user ID is 64 characters.

If the MQSeries server is on Windows NT, and the client is on a platform that uses the environment variable for specifying the user ID but no domain is specified, the MQSeries server attempts to retrieve user account information from its primary domain or trusted domains. In this case, the maximum length for the user ID is 20 characters.

| The MQSeries for Windows NT server does not support connection of an MQSeries
| for Windows NT client when the client is running under a user ID that contains
| the @ character, for example, abc@d. The return code to the **MQCONN** call at the
| client is MQRC_NOT_AUTHORIZED.

On all other platforms and configurations, the maximum length for user IDs is 12
characters.

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 MCA. 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:

* Any values set by security exits
* The user ID (Windows NT or UNIX clients) or MQ_USER_ID environment
  variable (other clients) from the client
* MCAUSER (in the server-connection channel definition)

Depending upon the combination of settings of the above, MCAUserIdentifier is set
to the appropriate value. If a server security exit is provided, MCAUserIdentifier
can be set by the exit. Otherwise MCAUserIdentifier is determined as follows:

* If the server-connection channel MCAUSER attribute is nonblank, this value is
  used.
* If the server-connection channel MCAUSER attribute is blank, the user ID
  received from the client is used. However, for the clients that use the
  MQ_USER_ID environment variable to supply the user ID, it is possible that no
  environment variable is set. In this case, the user ID that started the server
  channel is used.

  | For the Windows 95 and Windows 98 clients, if the MQ_USER_ID environment
  | variable has not been set, the current Windows logged-on user ID provides user
  | identification for the client. If the MQ_USER_ID environment variable has not
  | been set and the current user has not logged on to Windows, no client user
  | identification is provided to the server.

When the MCAUserIdentifier is derived from the user ID that started the server
channel, the following value is used:

* For OS/390, the user ID assigned to the channel initiator started task by the
  OS/390 started procedures table. See the *MQSeries for OS/390 System Management
  Guide* for more information.
* For TCP/IP (non-OS/390), the user ID from the inetd.conf entry, or the user ID
  that started the listener.
* For SNA (non-OS/390), the user ID from the SNA Server entry or (if there is
  none) the incoming attach request, or the user ID that started the listener.
* For NetBIOS or SPX, the user ID that started the listener.

If any server-connection channel definitions exist that have the MCAUSER
attribute set to blank, clients can use this channel definition to connect to the
queue manager with access authority determined by the user ID supplied by the
client. This might be a security exposure if the system on which the queue
manager is running allows unauthorized network connections. The MQSeries
default server-connection channel (SYSTEM.DEF.SVRCONN) has the MCAUSER

attribute set to blank. **To prevent unauthorized access**, update the MCAUSER attribute of the default definition with a user ID that has no access to MQSeries objects.

When you define a channel with `runmqsc`, the MCAUSER attribute is changed to uppercase unless the user ID is contained within single quotation marks.

For Windows NT and UNIX servers, the content of the user ID that is received from the client is changed to lowercase.

Chapter 8. Using channels

This chapter discusses the following topics:

- "What is a channel?"
- "Defining MQI channels" on page 120
- "Creating a queue manager and starting MQSC on the server (not OS/390)" on page 121
- "Creating one definition on the MQSeries client and the other on the server" on page 121
- "Creating both definitions on the server" on page 124
- "Channel exits" on page 127
- "Connecting a client to a queue-sharing group" on page 128
- "Migrating from MQSeries Version 2 products to Version 5" on page 128

What is a channel?

A channel is a logical communication link between an MQSeries client and an MQSeries server, or between two MQSeries servers. A channel has two definitions: one at the server end of the connection, and one at the receiver end. The same *channel name* must be used at each end of the connection, and the *channel type* used must be compatible.

There are two categories of channel in MQSeries, with different channel types within these categories:

Message Channels

A message channel is a one-way link. It connects two queue managers via *message channel agents* (MCAs). Its purpose is to transfer messages from one queue manager to another. Message channels are not required by the client server environment. More information on message channels can be found in the *MQSeries Intercommunication*
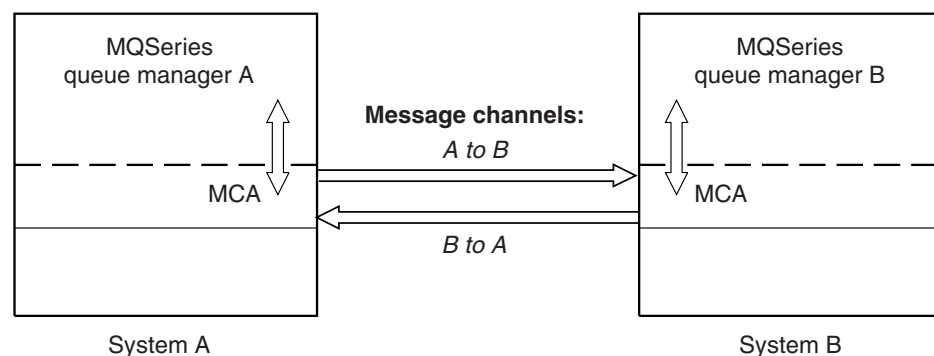

Figure 5. Message channels between two queue managers

MQI Channels

A message channel is a one-way link. It connects two queue managers via *message channel agents* (MCAs). Its purpose is to transfer messages from one queue manager to another.

An MQI channel connects an MQSeries client to a queue manager on a server machine, and is established when you issue an **MQCONN** or **MQCONNX** call. It is a two-way link and is used for the transfer of MQI calls and responses only, including **MQPUT** calls that contain message data. There are different ways of creating and using the channel definitions (see "Defining MQI channels").
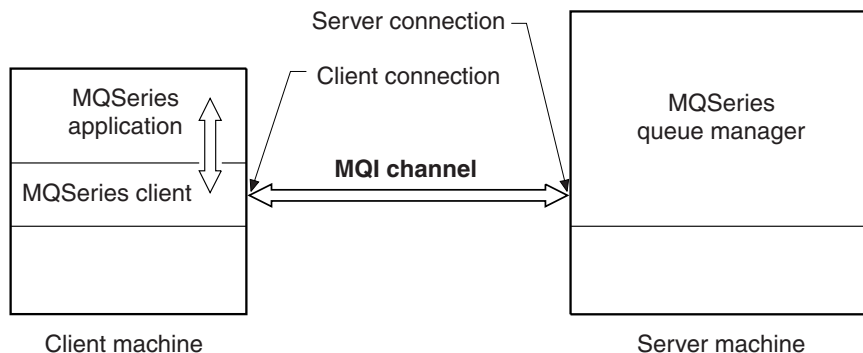
Figure 6. Client-connection and server-connection on an MQI channel

An MQI channel can be used to connect a client to a single queue manager, or to a queue manager that is part of a queue-sharing group (see "Connecting a client to a queue-sharing group" on page 128).

There are two channel types for MQI channel definitions. They define the bi-directional MQI channel.

Client-connection channel
This type is for the MQSeries client.

Server-connection channel
This type is for the server running the queue manager, with which the MQSeries application, running in an MQSeries client environment, will communicate.

Defining MQI channels

To create a 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*.

Automatically defined channels

The MQSeries Version 5.1 products 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 *MQSeries Intercommunication* book.

User defined channels

Where the server does not automatically define channels there are two 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 applies to any combination of MQSeries client and server platforms. Use it when you are getting started on the system, or to test your setup.

   See "Creating one definition on the MQSeries client and the other on the server" for details on how to use this method..

2. Create both channel definitions on the server machine.

   Use this method when you are setting up multiple channels and MQSeries client machines at the same time.

   See "Creating both definitions on the server" on page 124 for details on how to use this method.

Whichever method you choose, first you will need to create a queue manager and start MQSeries commands (see "Creating a queue manager and starting MQSC on the server (not OS/390)").

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. 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 OS/390, you first create and start a queue manager and then start MQSeries commands (MQSC).

Creating a queue manager and starting MQSC on the server (not OS/390)

1. create a queue manager, called QM1 for example:

```
crtmqm QM1
```

2. Start the queue manager:

```
strmqm QM1
```

3. Start MQSeries commands (MQSC)

Defining channels

On all platforms except VSE/ESA, start MQSC by entering the following command:

```
runmqsc QM1
```

Note: Creating MQM on Tandem requires extra mandatory parameters, See the *MQSeries for Tandem NonStop Kernel System Management Guide* for more details.

If you are using one of the MQSeries Version 5.1 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" on page 106.

MQSC is not supported on VSE/ESA. However, you can start MQSeries for VSE/ESA in any of the following ways:
- Use the MQSeries for VSE/ESA System Management transaction MQMT (panel 2.4).
- Use the transactions MQSE and MQIT.
- Automatically through CICS/VSE®® initialization.

See the *MQSeries for VSE/ESA System Management Guide* for more information.

Note: Two MQSeries administration tools are provided with MQSeries for Windows NT Version 5.1. The first is a graphical administration tool called the MQSeries Explorer. The MQSeries Explorer runs as a snap-in within the Microsoft Management Console (MMC) under Windows NT, and is installed as part of the server component of MQSeries for Windows NT. The second is a Web Administration tool which provides administrators with secure sessions through which they can administer an MQSeries network. For more information about these, see the *MQSeries System Administration* manual.

Define the server connection
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 126).

For example:

```
DEFINE CHANNEL(CHAN1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server-connection to Client_1')
```

On the MQSeries client
There are two ways of defining the client connection on the client machine:

Using MQSERVER
You can use a single environment variable, MQSERVER (see "Chapter 9. Using MQSeries environment variables" on page 131), without specifying all the attributes, to define a simple client-connection as part of your channel definition.
- Define a simple channel on DOS, OS/2 Warp, Windows 95, Windows 98, Windows NT, or Windows 3.1 as follows:
  ```
  SET MQSERVER=ChannelName/TransportType/ConnectionName
  ```

  Or, for OS/2 Warp using LU 6.2 only, as follows:

```
SET MQSERVER=ChannelName/LU62/ConnectionName/ModeName/TpName
```

- Define a simple channel on UNIX systems as follows:

```
export MQSERVER=ChannelName/TransportType/ConnectionName
```

Where:

- `ChannelName` must be the same name as defined on the server. It cannot contain a forward slash.
- `TransportType` can be one of the following, depending on your MQSeries client platform (see Table 1 on page 12):
  - LU62
  - TCP
  - NETBIOS
  - SPX
  - DECNET

  Note: On UNIX systems the TransportType is case sensitive and must be uppercase. An **MQCONN** or **MQCONNX** call will return 2058 if the TransportType is not recognized

- `ConnectionName` is the name of the server machine as defined to the communications protocol (TransportType).
- For OS/2 Warp, `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 *MQSeries Intercommunication* manual.

For example, on OS/2 Warp:

```
SET MQSERVER=CHAN1/TCP/MCID66499
```

or, on a UNIX system:

```
export MQSERVER=CHAN1/TCP/'MCID66499'
```

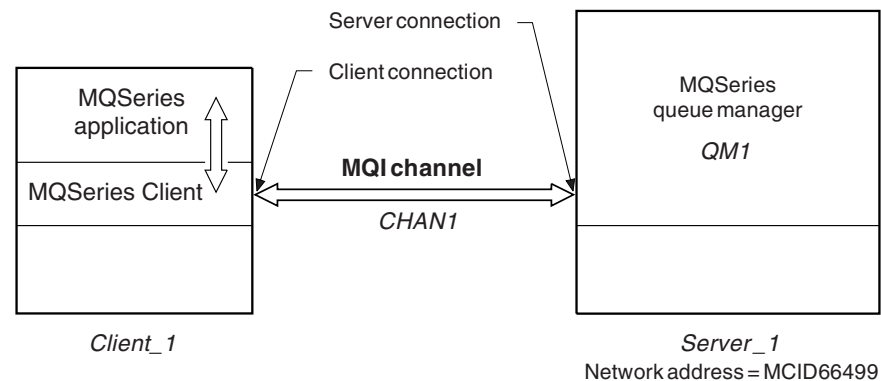**Note:** To change the TCP/IP port number, see "MQSERVER" on page 134.

Figure 7. Simple channel definition

Some more examples of defining the simple channel on DOS, OS/2 Warp, 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
```

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: For more information on the MQSERVER environment variable see "Chapter 9. Using MQSeries environment variables" on page 131.

Using the MQCNO structure in MQCONNX

If you are using one of the Version 5.1 clients, you can specify in a channel data (MQCD) structure the definition of the client-connection channel by using the MQCNO structure of the **MQCONNX** call issued by the client application (see "Using the MQCNO structure in MQCONNX").

This allows a calling client application to specify the ConnectionName attribute of the channel at run time, without using the MQSERVER environment variable. This enables an MQSeries client to connect to multiple servers simultaneously (which is not possible when you define the channel using the MQSERVER environment variable).

You can also specify attributes of the channel such as the MaxMsgLength and SecurityExit, allowing client applications that do not have a client channel table to specify non-default values for the channel attributes.

Note: This function is supported only in the Version 5.1 client environments. A sample connect program called amqscnxc demonstrates the use of this function. For more information, see the *MQSeries Application Programming Reference*.

Creating both definitions on the server

First define the server-connection channel and then define the client-connection channel.

On the server machine use MQSeries commands (MQSC) to define the channel.

Note: This method cannot be used on MQSeries for VSE/ESA.

Defining the server-connection channel

1. If your server platform is not OS/390, 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 (not OS/390)" on page 121
2. 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.
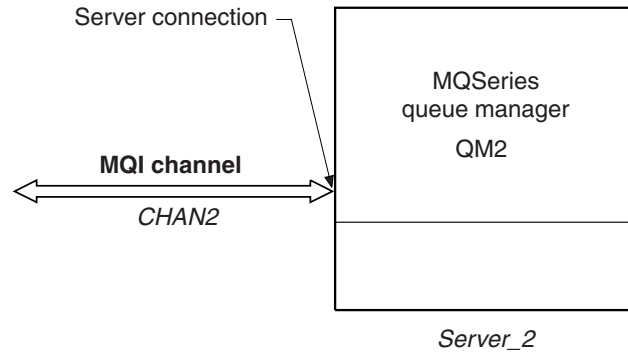

Figure 8. Defining the server-connection channel

Defining the client-connection channel

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 also advisable 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 151.

For example:

```
DEFINE CHANNEL(CHAN2) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) QMNAME(QM2) DESCR('Client-connection to Server_2')
```
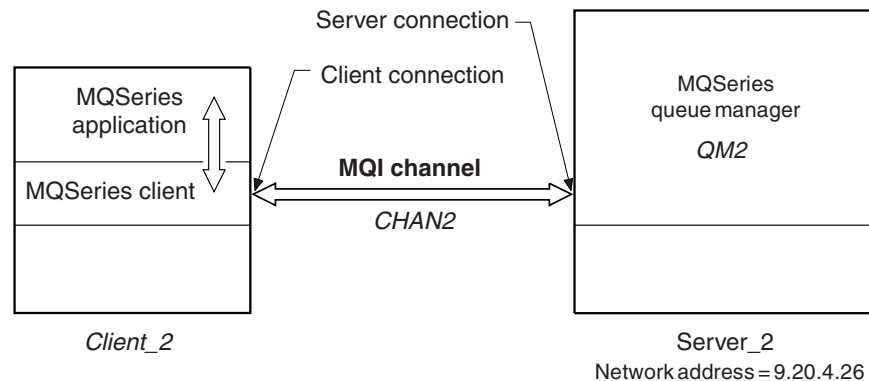

Figure 9. Defining the client-connection channel

Setting environment variables

On the MQSeries client machine, use the environment variables MQCHLLIB and MQCHLTAB to allow the MQSeries application to access the file containing the

Defining channels

client channel definition table on the server (but not a server on OS/390). See "MQCHLLIB" on page 132 and "MQCHLTAB" on page 133 for more information.

To set the environment variables on a UNIX system, type:

```
export MQCHLLIB=/var/mqm/qmgrs/QUEUEMANAGERNAME/@ipcc
export MQCHLTAB=AMQCLCHL.TAB
```

To set the environment variables on an OS/400 system, type:

```
ADDENVVAR ENVVAR(MQCHLLIB)
  VALUE('/QIBM/UserData/mqm/qmgrs/QUEUEMANAGERNAME/@ipcc')
ADDENVVAR ENVVAR(MQCHLTAB)
  VALUE('AMQCLCHL.TAB')
```

On OS/390, 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 *MQ Series for OS/390 System Management* book.

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 client channel definition pointed to by MQCHLLIB and MQCHLTAB will in some cases be overridden by the MQSERVER environment variable.

Client channel definition table

For non-OS/390 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.

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" on page 106.

AMQCLCHL.TAB is found in:
- For OS/2 Warp and Windows NT:

  `\mqm\qmgrs\queuemanagername\@ipcc`
- For Tandem NSK (The file is called CCHDEFS):

  `$volume.QMD`

  (Where QMD is the data subvolume of your queue manager.)
- For UNIX systems:

  /mqmtop/`qmgrs/QUEUEMANAGERNAME/@ipcc`

  `QUEUEMANAGERNAME` is case sensitive for UNIX systems.
- For OS/390 systems, the channel definition is kept with all other object definitions on page set zero.

Notes:

1. On Tandem NSK, a conversion utility (CNVCLCHL) is provided to convert the client channel definition table from a Tandem structured file to an unstructured one. See the *MQSeries for Tandem NonStop Kernel System Management Guide* for more details.

2. 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.

Channel exits

The channel exits available to the MQSeries client environment on OS/2 Warp, UNIX systems, Windows 3.1, Windows 95, Windows 98, and Windows NT are:
- Send exit
- Receive exit
- Security exit

Channel exits are not available on DOS systems.

These exits are available at both the client and the server end of the channel. Exits are not available to your application if you are using the MQSERVER environment variable. Exits are explained in the *MQSeries Intercommunication* manual.

The send and receive exit work together. There are several possible ways in which you can 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 Warp, Sun Solaris, Windows 95, and Windows 98 systems.

Channel exits

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.

For Windows NT this stanza is added to the Registry.

When a channel is initialized, after an **MQCONN** or **MQCONNX** call, the `mqs.ini` file, or Registry, is searched. The ClientExitPath stanza is read and any channel exits that are specified in the channel definition are loaded.

Migrating from MQSeries Version 2 products to Version 5

The internal format of the channel definition table has been changed in the MQSeries Version 5 products for AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT.

A Version 5 client will continue to work with a Version 2 client channel definition table (by default, the client channel definition table is called AMQCLCHL.TAB). However, a Version 2 client cannot read a Version 5 client channel definition table. If your client channel definition table is produced by a Version 5 server, you must use Version 5 clients. In this case, reinstall the MQSeries client from an MQSeries Version 5 server. In the case of a DOS MQSeries client, also relink your application.

Connecting a client to a queue-sharing group

MQSeries for OS/390 Version 5.2 supports queue-sharing groups.

You can connect a client to a queue-sharing group by creating an MQI channel between a client and a queue manager on a server machine that is a member of a queue-sharing group.

A queue-sharing group is formed by a set of queue-managers that can access the same set of shared queues. For more information on shared queues, see the *MQSeries for OS/390 Concepts and Planning Guide* book and the *MQSeries Intercommunication* book.

A client putting to a shared queue can connect to any member of the queue-sharing group. The benefits of connecting to a queue-sharing group are possible increases in front-end and back-end availablity, and increased capacity. You can connect a client to a queue-sharing group in the following ways.

Connecting to a specific queue manager

Connecting directly to a queue manager in a queue-sharing group gives the benefit that you can put messages to a shared target queue, which increases back-end availability.

Connecting to the generic address

Connecting to the generic address of a queue-sharing group will open a session with one of the queue managers in the group. This increases front-end availability, because the client queue manager can connect with any queue-manager in the

group. You connect to the group using the generic address when you do not want to connect to a specific queue manager within the queue-sharing group.

Creating channel definitions

To connect to the generic address of a queue-sharing group you need to create channel definitions that can be accessed by any queue manager in the group. To do this you need to have the same definitions on each queue manager in the group.

Define the SVRCONN channel as follows:

```
DEFINE CHANNEL(CHAN1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
MCAUSER(' ') QSGDISP(GROUP)
```

Channel definitions on the server are stored in a shared DB2® repository. Each queue manager in the queue-sharing group makes a local copy of the definition, ensuring that you will always connect to the correct server-connection channel when you issue an MQCONN or MQCONNX call.

Define the CLNTCONN channel as follows:

```
DEFINE CHANNEL(CHAN1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(generic address) QMNAME(QSG1) +
DESCR('Client-connection to Queue Sharing Group QSG1')
```

Because the generic address of the queue-sharing group is stored in the CONNAME field in the client-connection channel, you can now connect to any queue manager in the group, and put to shared queues owned by that group.

System administration

Chapter 9. Using MQSeries environment variables

This chapter describes the environment variables that you can use with MQI applications.

MQSeries uses default values for those variables that you have not set. Using environment variables, you can 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.

The MQSeries environment variables are:
- "MQCCSID" on page 132
- "MQCHLLIB" on page 132
- "MQCHLTAB" on page 133
- "MQDATA (DOS and Windows 3.1 only)" on page 133
- "MQNAME" on page 134
- "MQ_PASSWORD (DOS, OS/2 Warp, Windows 3.1, Windows 95, and Windows 98 only)" on page 134
- "MQSERVER" on page 134
- MQSPREFIX (used on the server, see the *MQSeries System Administration* manual)
- "MQTRACE (DOS, Windows 3.1, and VM/ESA only)" on page 136
- "MQ_USER_ID (DOS, OS/2 Warp, Windows 3.1, Windows 95, and Windows 98 only)" on page 136
- "MQSWORKPATH (OS/2 Warp only)" on page 136

Commands are available on all the MQSeries client platforms unless otherwise stated.

Notes:

1. MQSeries for OS/390 does not support *any* MQSeries environment variables. If you are using this platform as your server, see"MQCHLLIB" on page 132 for information about the location of the client channel definition table. You can still use the MQSeries environment variables on your client platform.

2. MQSeries for Tandem NonStop Kernel does not support *any* MQSeries environment variables. MQSeries for Tandem NonStop Kernel 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.

3. MQSeries for VSE/ESA does not support *any* MQSeries environment variables. You can still use the MQSeries environment variables on your client platform.

For each environment variable, use the command relevant to your platform to display the current setting or to reset the value of a variable.

Environment variables

For example:

Command	Effect
SET MQSERVER=	Removes the variable from DOS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, and Windows NT environments
unset MQSERVER	Removes the variable from UNIX systems environments
SET MQSERVER	Displays the current setting on DOS, OS/2 Warp, 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. See "Choosing client or server coded character set identifier (CCSID)" on page 139 for more information.

The format of this command is:
- For OS/2 Warp and Windows NT

  ```
  SET MQCCSID=number
  ```
- For UNIX systems

  ```
  export MQCCSID=number
  ```
- For VM/ESA

  ```
  GLOBALV SELECT CENV SETLP MQCCSID number
  ```

For more information, see the *MQSeries Application Programming Reference* manual.

MQCHLLIB

This holds the path to the directory and file containing the client channel definition table. The file containing the definition table is created on the server, but can be copied across to the MQSeries client machine. If MQCHLLIB is not set, the path defaults to:
- For DOS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, and Windows NT

  ```
  Rootdrive:\mqm\
  ```
- For UNIX systems

  ```
  /var/mqm/
  ```

If you are using MQSeries for OS/390 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 DOS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, and Windows NT

  ```
  SET MQCHLLIB=pathname
  ```
- For UNIX systems

  ```
  export MQCHLLIB=pathname
  ```

For example:

```
SET MQCHLLIB=C:\os2
```

Notes:

1. If you change the default setting of this variable *after* you create the queue manager, you must copy your existing client channel definition table to the new location.

2. If you change the default setting of this variable *before* you create the queue manager, you do not need to copy your existing client channel definition table to the new location.

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 following directory:

- For DOS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, and Windows NT

  `\mqm\qmgrs\`*queuemanagername*`\@ipcc`

- For UNIX systems

  `/var/mqm/qmgrs/`*QUEUEMANAGERNAME*`/@ipcc`

- For MQSeries for Compaq (DIGITAL) OpenVMS systems

  `mqs_root:[mqm.qmgrs.QM.$IPCC]`

The format of this command is:

- For DOS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, 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
```

MQDATA (DOS and Windows 3.1 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.

Environment variables

An error message is always added to the end of the log, so the files must be deleted periodically to avoid the files getting too large. If the file does not exist at the time a record needs to be added to one of these files 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 98 for a full description and for the rules of precedence on the client and the server.

The format of this command is:
```
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.

MQ_PASSWORD (DOS, OS/2 Warp, Windows 3.1, Windows 95, and Windows 98 only)

This specifies the password of the client. It is described in "Chapter 7. Setting up MQSeries client security" on page 115.

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. The *ChannelName* cannot contain the forward slash (/) character because it is used to separate the channel name, transport type, and connection name. When the MQSERVER environment variable is used to define a client channel a maximum message length (MAXMSGL) of 4 MB is used, so larger messages cannot flow across this channel. For larger messages a client-connection channel must be defined using DEFINE CHANNEL, on the server, with MAXMSGL set to a larger figure.

The format of this command is:
- For DOS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, and Windows NT
  ```
  SET MQSERVER=ChannelName/TransportType/ConnectionName
  ```
- For OS/2 Warp using LU 6.2
  ```
  SET MQSERVER=ChannelName/LU62/ConnectionName/ModeName/TpName
  ```
- For UNIX systems
  ```
  export MQSERVER=ChannelName/TransportType/ConnectionName
  ```
- For MQSeries for Compaq (DIGITAL) OpenVMS systems using DECnet
  ```
  define mqserver "ChannelName/decnet/nodename(object number)"
  ```

Or a symbol:

134 MQSeries Clients

```
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/ModeName/TpName
  ```

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 DOS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, 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 85.

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:
  ```
  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 Warp:

```
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
```

Examples on MQSeries for Compaq (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. However, if an MQCD structure has been defined in the MQCNO structure supplied to **MQCONNX**, the channel specified by the MQCD structure takes priority over any specified by the MQSERVER environment variable.

The MQSERVER environment variable takes priority over any client channel definition pointed to by MQCHLLIB and MQCHLTAB.

Cancelling MQSERVER

To cancel MQSERVER and return to the client channel definition table pointed to by MQCHLLIB and MQCHLTAB, enter the following:

- On DOS, OS/2 Warp, or Windows NT:

  `SET MQSERVER=`

- On a UNIX system:

  `unset MQSERVER`

MQTRACE (DOS, Windows 3.1, and VM/ESA only)

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`

MQ_USER_ID (DOS, OS/2 Warp, Windows 3.1, Windows 95, and Windows 98 only)

This specifies the user ID of the client. It is described in "Chapter 7. Setting up MQSeries client security" on page 115.

MQSWORKPATH (OS/2 Warp only)

This specifies the path to the mqs.ini file and is used internally by MQSeries.

Part 4. Application programming

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 discusses the following topics:
- "Limiting the size of a message"
- "Choosing client or server coded character set identifier (CCSID)"
- "Controlling application in a Windows 3.1 environment" on page 140
- "Designing applications" on page 140
- "Using MQINQ" on page 140
- "Using syncpoint coordination" on page 141
- "Using MQCONNX" on page 141

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.

On MQSeries Version 5.1 products, you can increase the maximum message length attribute of a queue manager. 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 coded character set identifier (CCSID), encoded for the MQSeries client. If the connected queue manager requires the data to be converted, this is done by the client support code.

The client code assumes that the character data crossing the MQI in the client is in the CCSID configured for that machine. 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.

CCSID and encoding fields - multiple puts

If your application is performing multiple PUTs that include MQSeries headers after the message descriptor (MQMD), be aware that the CCSID and encoding fields of the MQMD are overwritten after completion of the first PUT. After the first PUT, these fields contain the value used by the connected queue manager to convert the MQSeries headers. Ensure that your application resets the values to those it requires.

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 receives 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: The only way that an ongoing MQI call can be cancelled is by the application receiving a WM_QUIT message.

Using MQINQ

Some values queried using **MQINQ** are 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

For more information, see the *MQSeries Application Programming Guide*.

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, 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 5.1 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 NonStop Kernel server

When an MQSeries client connects to a queue manager on MQSeries for Tandem NonStop Kernel, V2.2.0.1:

- 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 the following options are ignored:
    MQCNO_STANDARD_BINDING
    MQCNO_FASTPATH_BINDING

MQCONNX and **MQCONN** on a client are similar calls, except that **MQCONNX** allows a client application to specify a channel data (MQCD) structure in the MQCNO structure. This allows the calling client application to specify the definition of the client-connection channel at run time (see "Using the MQCNO structure in MQCONNX" on page 124). The actual call issued at the server depends on the server level and the listener configuration.

Using the MQI

Chapter 11. Building applications for MQSeries clients

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.

It discusses the following topics:
- "Running applications in the MQSeries client environment" on page 144
- "Triggering in the client environment" on page 144
- "Channel exits" on page 146
- "Linking C applications with the MQSeries client code" on page 147
- "Linking C++ applications with the MQSeries client code" on page 149
- "Linking COBOL applications with the MQSeries client code" on page 149
- "Linking PL/I applications with the MQSeries client code" on page 150
- "Linking Visual Basic applications with the MQSeries client code" on page 150

If an application is to run in a client environment, you can write it in the languages shown in the following table:

Table 5. Programming languages supported in client environments

Client platform	Assembler	C	C++(1)	COBOL	Java(2)	PL/I	REXX	RPG	Visual Basic
AIX		✔	✔	✔	✔	✔			
AT&T GIS UNIX		✔							
Compaq Tru64 UNIX		✔	✔	✔	✔				
MQSeries for Compaq (DIGITAL) OpenVMS		✔		✔					
DOS		✔							
HP-UX		✔	✔	✔	✔				
OS/2 Warp		✔	✔	✔	✔	✔			
SINIX and DC/OSx		✔		✔					
Sun Solaris		✔	✔	✔	✔				
Sun Solaris, Intel Platform Edition		✔	✔	✔	✔				
VM/ESA	✔	✔		✔		✔	✔		
Windows 3.1		✔	✔	✔					✔
Windows 95		✔	✔	✔					✔
Windows 98		✔	✔	✔					✔
Windows NT		✔	✔	✔	✔	✔			✔

Note:

1. If you are using an MQSeries client supplied with an MQSeries Version 5 product, you can write applications to run on the client in C++. Programs that use the MQSeries C++ classes can be used successfully with MQSeries Version 5 and MQSeries for OS/390 servers only. To see how to link your C++ applications and for information about all aspects of using C++ see the *MQSeries Using C++* manual.

2. If you have installed the MQSeries client for Java, you can write Java applets that communicate with MQSeries. Remember to link your application to the relevant client library file. For information, see the *MQSeries Using Java* manual.

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, provided that:

- 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 that :

- 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 132.

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.

The default characteristics of the triggered queue are the same as those in the server environment. In particular, if no MQPMO syncpoint control options are specified in a client application putting messages to a triggered queue that is local to an OS/390 queue manager, the messages are put within a unit of work. If the triggering condition is then met, the trigger message is put on the initiation queue within the same unit of work and cannot be retrieved by the trigger monitor until the unit of work ends. The process that is to be triggered is not started until the unit of work ends.

Process definition

You must define the process definition on the server, because 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 Warp server, MQAT_WINDOWS_NT must be defined otherwise OS/2 Warp 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.

MQSeries client for Windows 95 and Windows 98

The MQSeries client for Windows 95 and Windows 98 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 or Windows 98 platform. If a trigger monitor is running on a Windows 95 and Windows 98 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-OS/390 MQSeries products runs in the MQSeries client environments for MQSeries for Compaq (DIGITAL) OpenVMS, OS/2 Warp, Windows 3.1, Windows 95, Windows 98, Windows NT, and UNIX systems. To run the trigger monitor, 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 that 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 does 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 might mean that the trigger monitor cannot keep up with the number of trigger messages arriving. There are two ways to deal with this:

1. Have more trigger monitors running

   If you choose to have more trigger monitors running, you can control the maximum number of applications that can run at any one time.
2. Run the started applications in the background

   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 Warp 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-OS/390)

A non-OS/390 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, you can link 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 on OS/2 Warp, UNIX systems, Windows 3.1, Windows 95, Windows 98, and Windows NT are:
- Send exit
- Receive exit
- Security exit

Channel exits are not available on DOS systems.

These exits are available at both the client and the server end of the channel. Exits are not available to your application if you are using the MQSERVER environment variable. Exits are explained in the *MQSeries Intercommunication* manual.

The send and receive exit work together. There are several possible ways in which you can 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 Warp, Sun Solaris, Windows 95, and Windows 98 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.

For Windows NT this stanza is added to the Registry.

When a channel is initialized, after an **MQCONN** or **MQCONNX** call, the `mqs.ini` file, or Registry, 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:

1. Directly, in which case the queue manager must be on the same machine as your application

2. 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 non-threaded applications, or libmqic_r.a library for 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 -lmqmzse -lnet \
>   -lnsl -lsocket -ldl -lc
> ```

MQSeries for Compaq (DIGITAL) OpenVMS
> MQIC.EXE in SYS$SHARE.

Compaq Tru64 UNIX
> libmqic.so libraryfor Version 4 non-threaded applications, libmqic_r.so for Version 4 threaded applications, or libmqic.so for Version 5 applications.

DOS   MQIC.LIB.

> 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.

HP-UX
> libmqic.sl.

OS/2 Warp
> 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
> ```
>
> The parameters must be entered in the correct order, as shown.

Sun Solaris, Intel Platform Edition
> libmqic.so, libmqmcs.so, or libmqmzse.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 -lmqmzse -lsocket -lc -lnsl -ldl
> ```
>
> The parameters must be entered in the correct order, as shown.

Windows 3.1
> LIBW.LIB, LLIBCEW.LIB, MQIC.LIB.

Windows NT
> MQIC32.LIB.

Windows 95 and Windows 98
> MQIC32.LIB.

Running 16–bit and 32-bit Windows clients

Previous versions of the MQSeries clients for Windows NT and Windows 95 and Windows 98 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–bit and 32-bit environments.

Linking C++ applications with the MQSeries client code

If you are using an MQSeries client supplied with an MQSeries Version 5.1 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 V5.1 and OS/390 V2.1 servers only. For information about how to link your C++ applications and for full details of all aspects of using C++, see the *MQSeries Using C++* manual.

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.

Compaq Tru64 UNIX
Link your COBOL application with the lmqmicb library for non-threaded applications, or lmqmicb_r library for threaded applications..

MQSeries for Compaq (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 depends on how you are using the -B flag during the linking stage. For more information see the *MQSeries Application Programming Guide*.

OS/2 Warp
If you have an OS/2 Warp 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 might 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.

Linking applications

Sun Solaris (SPARC and Intel Platform Edition)
Link your COBOL application with the libmqicb.so library.

Windows NT, Windows 95, and Windows 98
If you have a Windows NT, Windows 95, or Windows 98 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 client for Windows 95 and Windows 98 do not support 16-bit COBOL.

Linking PL/I applications with the MQSeries client code

AIX Link your PL/I application with libmqic.a library for threaded applications, or libmqic_r.a library for non-threaded applications.

OS/2 Warp
Link your PL/I application with the MQIC.LIB library.

Windows NT
Link your PL/I application with the MQIC32.LIB library.

See the *MQSeries Application Programming Guide* for further details.

Linking Visual Basic applications with the MQSeries client code

You can link Visual Basic applications with the MQSeries client code on Windows 3.1, Windows 95, Windows 98, and Windows NT.

Link your Visual Basic application with the following include files:
CMQB.bas
MQI
CMQBB.bas
MQAI
CMQCFB.bas
PCF commands
CMQXB.bas
Channels

Set `mqtype=2` for the client in the Visual Basic compiler, to ensure the correct automatic selection of the client dll:
MQIC.dll
Windows 3.1
MQIC32.dll
Windows 95 and Windows 98
MQIC32.dll
Windows NT

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 following topics:

- "Using environment variables" on page 152
- "Using the MQCNO structure" on page 152
- "Using DEFINE CHANNEL" on page 152
- "Role of the client channel definition table" on page 152
- "Examples of MQCONN calls" on page 153

When an application running in an MQSeries client environment issues an **MQCONN** or **MQCONNX** call, the client identifies how it is to make the connection. When an **MQCONNX** call is issued by an application on a Version 5.1 client, the MQI client library searches for the client channel information in the following order:

1. Using the contents of the *ClientConnOffset* or *ClientConnPtr* fields of the MQCNO structure (if supplied). These identify the channel data structure (MQCD) to be used as the definition of the client connection channel.

2. If the MQSERVER environment variable is set, the channel it defines is used.

3. If the MQCHLLIB and MQCHLTAB environment variables are set, the client channel definition table they point to is used.

4. Finally, if the environment variables are *not* set, the client searches for a channel definition table whose path and name are established from the `DefaultPrefix` in the `mqs.ini` file or the Registry for Windows NT. If this fails, the client uses the following paths:

   - MQSeries for Compaq (DIGITAL) OpenVMS

     `mqs_root:[mqm]amqclchl.tab`

   - OS/2 Warp

     `MQSWORKPATH\amqclchl.tab`

     Or, if MQSWORKPATH is not set:

     `bootdrive:\mqm\amqclchl.tab`

   - UNIX systems

     `/var/mqm/AMQCLCHL.TAB`

   - Windows NT, Windows 95, and Windows 98

     `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`.

The first of the options described above (using the *ClientConnOffset* or *ClientConnPtr* fields of MQCNO) is supported only by the **MQCONNX** call when running on a Version 5.1 client. If the application is running on a non-Version 5.1 client, or if it is using **MQCONN** rather than **MQCONNX**, the channel information is searched for in the remaining three ways in the order shown above. If the client fails to find any of these, the **MQCONN** or **MQCONNX** call fails.

The channel name (for the client connection) must match the server-connection channel name defined on the server for the **MQCONN** or **MQCONNX** call to succeed.

If you receive an 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 Version 2 products to Version 5" on page 128.

Using environment variables

Client channel information can be supplied to an application running in a client environment by the MQSERVER, MQCHLLIB, and MQCHLTAB environment variables.

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 MQCHLLIB and MQCHLTAB

The MQCHLLIB environment variable on the server specifies the path to the directory containing the client channel definition table. The MQCHLTAB environment variable (also on the server) specifies the name of the client channel definition table. For more information about these environment variables, see "Chapter 9. Using MQSeries environment variables" on page 131.

Using the MQCNO structure

You can specify the definition of the channel in a channel data structure (MQCD), which is supplied using the MQCNO structure of the **MQCONNX** call. For more information see "Using the MQCNO structure in MQCONNX" on page 124.

Using DEFINE CHANNEL

If you use the MQSC DEFINE CHANNEL command, the details you provide are placed in the client channel definition table. This file is accessed by the client, 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 "Defining MQI channels" on page 120 for more information about what you have to define and how you do it.

Note: The same file can 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 9. Using MQSeries environment variables" on page 131 for information about environment variables.

Multiple queue managers

You might 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.

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.

Queue-sharing groups

You may wish to connect your application to a queue manager that is part of a queue-sharing group. This can be done by issuing the following call:

```
MQCONN(QSG1)
```

Where QSG1 is the name of the queue-sharing group.

The client channel definition should use the queue sharing group generic address to connect to an available queue manager in the group. A check is made to ensure that the queue manager the listener connects to is a member of the queue sharing group.

For more information on shared queues, see the see the *MQSeries for OS/390 Concepts and Planning Guide* book, and the *MQSeries Intercommunication* book.

Examples of MQCONN calls

In each of the following examples, the network is the same; there is a connection defined to two servers from the same MQSeries client. (In these examples, the **MQCONNX** call could be used instead of the **MQCONN** call.)

There are two queue managers running on the server machines, one named SALE and the other named SALE_BACKUP.

Running applications
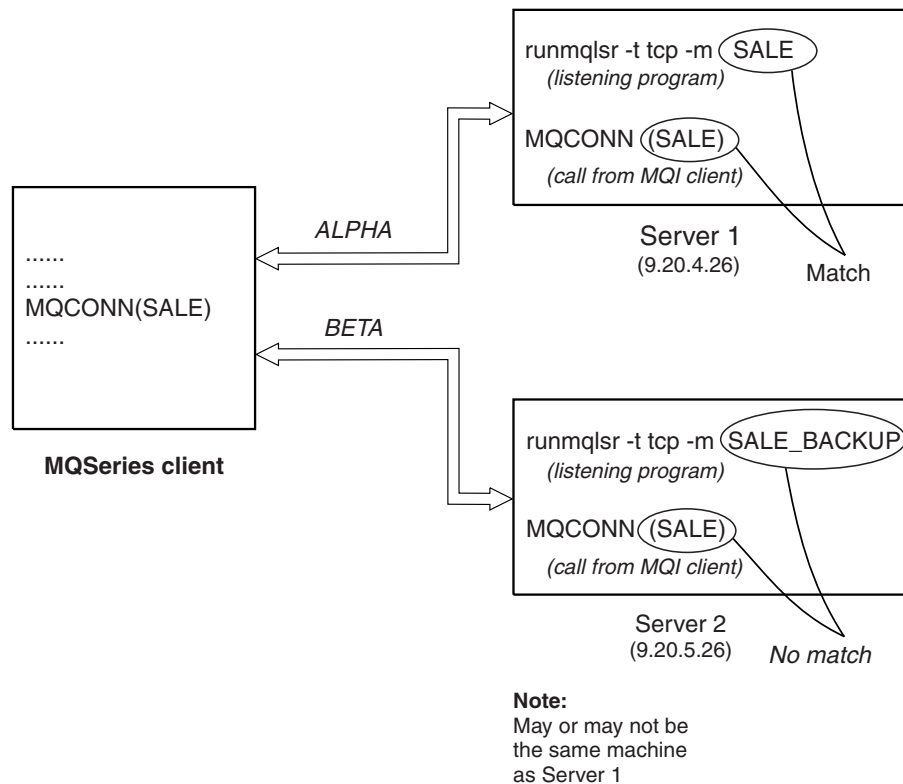
Figure 10. MQCONN example

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 can be summarized as follows:

Name	CHLTYPE	TRPTYPE	CONNAME	QMNAME
ALPHA	CLNTCONN	TCP	9.20.4.26	SALE
BETA	CLNTCONN	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 continues. 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 includes 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 happens 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 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 (*) included in it, no check is made. The application is connected to the SALE_BACKUP queue manager and continues 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 happens 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 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 receives return code MQRC_Q_MGR_NOT_AVAILABLE.

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 happens 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 receives return code MQRC_Q_MGR_NOT_AVAILABLE.

Chapter 13. Solving problems

This chapter discusses the following topics:

- "MQSeries client fails to make a connection"
- "Stopping MQSeries clients" on page 158
- "Error messages with MQSeries clients" on page 158
- "How to read the error log and FFDCs for DOS and Windows 3.1" on page 158
- "Using trace on DOS and Windows 3.1" on page 159
- "Using trace on OS/2 Warp, Windows NT, Windows 95, and Windows 98" on page 160
- "Using trace on AIX and AT&T GIS UNIX" on page 161
- "Using trace on MQSeries for Compaq (DIGITAL) OpenVMS, HP-UX, SINIX, DC/OSx, and Sun Solaris" on page 162
- "Using trace on VM/ESA" on page 163

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 might 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 is not functioning correctly, 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 exchanged and further checking takes place. During the MQSeries protocol checking phase, some aspects are negotiated while others cause the connection to fail. It is not until all these checks are successful that the **MQCONN** or **MQCONNX** call succeeds.

For information about the MQRC_* reason codes, see the *MQSeries Application Programming Reference* manual.

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 attempts to place the error message in an error log in the
root directory of the MQSeries client machine.

MQSeries for Compaq (DIGITAL) OpenVMS, OS/2 Warp, UNIX systems, Windows 95, Windows 98, and Windows NT

Error messages for MQSeries clients on MQSeries for Compaq (DIGITAL)
OpenVMS, OS/2 Warp, UNIX systems, Windows 95, Windows 98, and Windows
NT are placed in the error logs in the same way they are for the respective
MQSeries server systems. Typically these files appear in:

* The MQS_ROOT:[MQM.ERRORS] directory for MQSeries for Compaq
  (DIGITAL) OpenVMS systems
* /var/mqm/errors for UNIX systems
* /mqm/errors for OS/2 Warp, Windows 95, Windows 98, and Windows NT
  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 9. Using MQSeries
environment variables" on page 131 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 file name 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 because 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.

Using trace on DOS and Windows 3.1

Use the MQTRACE environment variable to set tracing on (see "MQTRACE (DOS, Windows 3.1, and VM/ESA only)" on page 136). 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 file name for each of the applications to guarantee that no trace entries are lost.

Figure 11 on page 160 shows an example.

Example DOS trace data

The following example shows an extract from a trace for a DOS MQSeries client:

Trace

```
Trace started on Mon Oct 19 10:48:42 1998
PC 750-P133 - 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 11. Extract from a DOS client trace

The entries in the box represent data sent or received over communications links.

Using trace on OS/2 Warp, Windows NT, Windows 95, and Windows 98

MQSeries for Windows NT and MQSeries for OS/2 Warp 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 or Windows 98 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 is one trace file for each process running as part of the entity being traced.

How to examine First Failure Support Technology™ (FFST) files

The files are produced already formatted and are in the `\mqm\errors` directory.

Note: For V.5 clients on OS/2, Windows 95 and Windows NT, the errors directory is not created automatically during installation. The errors directory should be created before starting the trace.

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* manual.

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 might 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.

Trace

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/test/first.dat
```

When you want to stop, type **trcstop** to end the trace. Then format the trace file with the command:

```
trcrpt -t /usr/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* manual and the *MQSeries for AT&T GIS UNIX System Management Guide*.

Using trace on MQSeries for Compaq (DIGITAL) OpenVMS, HP-UX, SINIX, DC/OSx, and Sun Solaris

MQSeries for MQSeries for Compaq (DIGITAL) OpenVMS, HP-UX, SINIX and DC/OSx, 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* manual for Version 5.1 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 MQSeries for Compaq (DIGITAL) OpenVMS systems, the files are in the MQS_ROOT:[MQM.ERRORS] directory.

All queue manager tracing, all early tracing, and all @SYSTEM tracing takes place to files in this directory.

You can handle large trace files by mounting a temporary file system 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 is one trace file for each process running as part of the entity being traced.

How to examine FFSTs

Information that is normally recorded in FFST logs on OS/2 Warp and AIX is recorded in a file in the /var/mqm/errors directory on AT&T GIS UNIX, HP-UX, SINIX, DC/OSx, and Sun Solaris.

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* manual for MQSeries Version 5.1 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 134). 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 10/09/95 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 12. Extract from a VM/ESA client trace

Part 5. Appendixes

Appendix A. Installation response file format for Windows NT

The installation response file has a standard Windows .ini file format. All text is in U.S. English, lines beginning with a semicolon (;) are comments, and case is ignored.

Sample installation response file

The following is a sample response file. It consists of stanzas with titles in square brackets, and, within each stanza, parameters in keyword=value format. Four stanzas are required:
- [InstallShield Silent]
- [DlgOrder]
- [Application]
- [MQSeries-0]

Of these four, only the [MQSeries-0] stanza specifies the installation parameters. The remaining three should be left unchanged or coded as shown.

```
[InstallShield Silent]
Version=v5.00.000
File=Response File
[Application]
Name=MQSeries
Version=CurrentVersion
Company=IBM
[DlgOrder]
Dlg0=MQSeries-0
Count=1
[MQSeries-0]
Default=NO
PgmDir=e:\mqmpgm
DatDir=e:\mqmdat
ProgramFolder=IBM MQSeries Client
Skip=NO
LockedFiles=CONTINUE
Development Toolkit=INSTALL
Local Clients\Windows NT Client=REMOVE
Local Clients\Java Client=REMOVE
Internet Gateway=REMOVE
Documentation\Internet Gateway=REMOVE
```

The [MQSeries-0] stanza

The Default parameter is required and can be used to specify a default installation. Its value can be NO, YES, or CURRENT.

Default=NO
> This value means that the installation is specified by the other keywords. You must code this value if you do not require one of the other values (YES or CURRENT).

Default=YES
> The installation updates the target machine to the latest MQSeries level. No new components are installed. If there is no existing MQSeries installation on the target machine, Setup installs all the client components. Other keywords in this stanza are ignored (and can be omitted).

Installation response file format

Default=CURRENT
> Interchangeable with Default=YES.

PgmDir=<folder>
> For a new installation, this value specifies the top-level folder for program files. The value <folder> must be a valid path on the target machine, or it can be DEFAULT. The value DEFAULT tells Setup to take the default value for the folder on the target machine, which is usually `c:\program files\MQSeries`. If there is a previous MQSeries installation on the machine, Setup continues to use the existing folder irrespective of the value of this parameter. For example:
>
> `PgmDir=c:\mqm pgm or PgmDir=default`

DatDir=<folder>
> For a new installation, this value specifies the top-level folder for data files. The value <folder> must be a valid path on the target machine, or it can be DEFAULT. The value DEFAULT tells Setup to take the default value for the folder on the target machine, which is usually `c:\Winnt\Profiles\All Users\Application Data\MQSeries`. If there is a previous MQSeries installation on the machine, Setup continues to use the existing folder irrespective of the value of this parameter. For example:
>
> `DatDir=c:\mqm data or DatDir=default`

Folder=<programfolder>
> For a new installation, this value specifies the program folder name for MQSeries. The value <programfolder> must be a valid program folder name, or it can be DEFAULT. The value DEFAULT tells Setup to take the default value for the program folder on the target machine, which is IBM MQSeries Client. If there is a previous MQSeries installation on the machine, Setup continues to use the existing program folder irrespective of the value of this parameter. For example:
>
> `ProgramFolder=My MQ Programs or ProgramFolder=default`

Skip=<option>
> Specifies whether or not to reinstall any components that are already installed and are at the Version 5.1 level. The value of <option> can be YES or NO. For example:
>
> `Skip=YES`

LockedFiles=<option>
> Before transferring files to the target computer, Setup checks to see if any of the files in needs to replace are locked. This option specifies what Setup will to do if it finds any such files.
>
> **CONTINUE**
> > Ignore the locked files and continue. After file transfer, Setup will restart the computer and replace the files during the restart.
>
> **CANCEL**
> > Terminate the installation before transferring any files.
>
> **NORESTART**
> > Ignore the locked files and continue. If any files are still found to be locked during file transfer, do not restart the computer. In this event you must restart the computer manually for correct MQSeries operation.

The other keywords in this stanza specify component selections. There must be one keyword=value pair for each component. The keyword is the component name and the value must be one of:

INSTALL
>   Install or reinstall the component.

REMOVE
>   Remove the component if it is already installed.

For example:

- `Windows NT Client=INSTALL` installs the Windows NT Client component, or if it is already installed, reinstalls it.
- `Development Toolkit=REMOVE` does not install the Development Toolkit component, or if it is already installed, removes it.

The component names are listed in "Component names used in response files".

Component names used in response files

The component names used in response files have fixed U.S. English values that are the same as the values stored in the registry after installation, under the following key:

`HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Components`

Their names are:
- Development Toolkit
- Local Clients\Windows NT Client
- Local Clients\Java Client
- Internet Gateway
- Documentation\Internet Gateway

Note that the names displayed during an attended installation are slightly different from these and are translated into the installation language.

Application programming

Appendix B. Uninstalling on Windows NT

You can uninstall (remove) MQSeries for Windows NT in attended mode or unattended (silent) mode.

Attended uninstallation

You can start attended uninstallation in one of the following ways:

- Start Uninstallation from the MQSeries program folder:
  1. Open the MQSeries program folder created during installation (by default this folder is called IBM MQSeries).
  2. Select MQSeries Uninstallation icon to run the uninstall program.
- Start Uninstallation by selecting the MQSeries entry from the Add/Remove programs icon in the Control Panel.
- Run the uninstall program directly from a command prompt or by using Run on the Start menu. Using this method, you can pass an execution parameter to the program to cause it to create and keep an uninstallation log file. The parameter is `-g<logfilepath>`, where `<logfilepath>` is the full path name of the log file you want to create.

  The uninstall program is located in the c:\program files\mqseries\uninst folder and called amqiunin.exe (where c:\program files\mqseries is the top-level program files folder selected during installation).

  For example, the command might be:

  ```
  c:\program files\mqseries\uninst\amqiunin -gc:\uninst.txt
  ```

  Without this parameter, uninstallation appends to or creates a log file called amqilogn.txt in the data files top-level folder. If you delete the data files folder during uninstallation, the log file is lost.

When the uninstallation program is running, you must select one of:
- Uninstall one or more components
- Uninstall all of MQSeries, excluding data
- Uninstall all of MQSeries, including data

Uninstalling MQSeries excluding data leaves your data files top-level directory intact. If you elect to uninstall one or more components, you must choose from a list of which components to uninstall.

Unattended uninstallation

You can uninstall the MQSeries for Windows NT client on a remote machine without interaction. This process is called unattended (or silent) uninstallation, and uses a response file.

A response file is an ASCII text file containing values for the options you select when you uninstall an MQSeries for Windows NT client system.

For unattended uninstallation, you can either:
- Edit the response file (amqiunin.rsp) supplied on the MQSeries for Windows NT client CD-ROM, using an ASCII file editor.

Uninstalling on Windows NT

- Generate your own response file using an ASCII file editor.

When you have generated your response file, run the uninstall program amqiunin.exe from the uninst directory in your chosen program-files folder as follows:

```
c:\program-files\mqseries client\uninst\amqiunin -g<logfile>
  -f1<responsefile> -s
```

where
- `<logfile>` is the fully-qualified path to an installation log file. You should check this log file to see if any errors have occurred. If you omit this parameter, a log file called `amqilogn.txt` is created in the data-files folder on the machine running the uninstallation; this will be deleted if the uninstall is successful.
- `<responsefile>` is the fully-qualified path to your response file. If this parameter is omitted, the response file must be called amqiunin.rsp and be located in the same folder as amqiunin.exe.
- The -s parameter tells the uninstall program to run in silent mode.

Uninstallation response file format

Like the installation response file, the uninstallation response file consists of stanzas and keyword=value pairs. Lines beginning with a semicolon (;) are comments. All text is in U.S. English and case is ignored. An example of a complete response file is shown below. Two stanzas are required: [MQSeries] and [Components].

The MQSeries stanza

This stanza has two required keywords: MQSeries and LockedFiles.

MQSeries keyword

MQSeries=REMOVEDATA
This causes uninstallation to remove all of the MQSeries client on the target machine. The [Components] stanza is not referenced and can be omitted.

MQSeries=REMOVE
For MQSeries client silent uninstallation, this keyword is the same as REMOVEDATA.

MQSeries=REMOVECOMPONENTS
This causes uninstallation to remove specified components. The components are listed in the [Components] stanza, which must be present.

LockedFiles keyword

LockedFiles=<option>
Before deleting files from the target computer, uninstallation checks to see if any of the files are locked. This parameter specifies what uninstallation is to do if it finds any locked files:

CONTINUE
Ignore the locked files and continue. You must remove any locked files manually after uninstallation.

CANCEL
Terminate the uninstallation before deleting any files.

The Components stanza

There can be one keyword=value pair in this stanza for each MQSeries for Windows NT client component. The keyword is the component name and the value must be KEEP, SKIP, REMOVE, or REMOVEDATA. The component names are the same strings that are used in an installation response file (see "Component names used in response files" on page 169). Any installed components for which there is no keyword=value pair are not uninstalled. For example:

```
Windows NT Client=REMOVEDATA or Windows NT Client=KEEP
```

Use the value REMOVE or REMOVEDATA when you want to remove a component. Use the value KEEP or SKIP when you do not want to remove a component. Figure 13 shows an example of a complete uninstallation response file:

```
[MQSeries]
MQSeries=REMOVECOMPONENTS
LockedFiles=CONTINUE
[Components]
Development Toolkit=REMOVE
Local Clients\Windows NT Client=KEEP
Local Clients\Java Client=KEEP
Internet Gateway=KEEP
Documentation\Internet Gateway=KEEP
```

Figure 13. Example Windows NT client uninstallation response file

In this example:
- The Windows NT components are to be removed.
- The Development Toolkit is removed.
- All other components are left unchanged.

The component names used in response files have fixed English values that are the same as the values stored in the registry after installation, under the following key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Components
```

The component names are:
- Development Toolkit
- Documentation/Internet Gateway
- Internet Gateway
- Local Clients/Java Client
- Local Clients/Windows NT Client

Note: The names presented to the user during an attended installation are slightly different from these, and are translated into the installation language.

Uninstalling on Windows NT

Appendix C. Notices

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

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:
    IBM Director of Licensing
    IBM Corporation
    North Castle Drive
    Armonk, NY 10504-1785
    U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
    IBM World Trade Asia Corporation
    Licensing
    2-31 Roppongi 3-chome, Minato-ku
    Tokyo 106, Japan

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

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

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

175

Notices

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

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

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

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

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

Trademarks

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

AIX	AS/400	AT
BookManager	CICS	CICS/VSE
DB2	First Failure Support Technology	FFST
IBM	IBMLink	IMS
Integrated Language Environment	Language Environment	MQSeries
MVS/ESA	OS/2	OS/390
OS/400	RACF	SupportPac
SP2	VM/ESA	VSE/ESA
WIN-OS/2		

NetView is a trademark of Tivoli Systems Inc. in the United States, or other countries, or both.

LotusScript is a trademark of Lotus Development Corporation in the United States, or other countries, or both.

Intel and Pentium are registered trademarks of Intel Corporation in the United States and/or other countries.

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

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

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

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

Application programming

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 OS/390 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*.

cluster. A network of queue managers that are logically associated in some way.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

command. In MQSeries, an administration 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 AS/400, 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 or MQCONNX 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).

Glossary

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

ILE. Integrated Language Environment®.

Integrated Language Environment® (ILE). The AS/400 Integrated Language Environment. This replaces the AS/400 Original Program Model (OPM).

.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, to enable them to recover in the event of failure.

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. In message queuing applications, a communication sent between programs. 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 at one end and a receiver at the other end) 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. See also *message queue interface*.

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*.

MQAI. MQSeries Administration Interface.

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 Administration Interface (MQAI). A programming interface to MQSeries.

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.

N

namelist. An MQSeries object that contains a list of names, for example, queue names.

O

object. In MQSeries, an object is a queue manager, a queue, a process definition, a channel, a namelist, or a storage class (OS/390 only).

object handle. The identifier or token by which a program accesses the MQSeries object with which it is working.

OPM. Original Program Model.

Original Program Model (OPM). The AS/400 Original Program Model. This is no longer supported on MQSeries. It is replaced by the Integrated Language Environment (ILE).

OTMA. Open Transaction Manager Access.

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. 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*. An MQSeries object that defines the attributes of a particular queue manager.

queue-sharing group. In MQSeries for OS/390, a group of queue managers in the same sysplex that can access a single set of object definitions stored in the shared repository, and a single set of shared queues stored in the coupling facility. See also *shared queue*.

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 OS/390, examples of resources are buffer pools, page sets, log data sets, queues, and messages.

Glossary

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*.

shared queue. In MQSeries for OS/390, a type of local queue. The messages on the queue are stored in the *coupling facility* and can be accessed by one or more queue managers in a *queue-sharing group*. The definition of the queue is stored in the *shared repository*.

shared repository. In MQSeries for OS/390, a shared DB2 database that is used to hold object definitions that have been defined globally.

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.

Bibliography

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, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for SINIX and DC/OSx, V2.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Sun Solaris, Intel Platform Edition, V5.1
- MQSeries for Tandem NonStop Kernel, V2.2.0.1
- MQSeries for VSE/ESA V2.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT, V5.1

The MQSeries cross-platform publications are:

- *MQSeries Brochure*, G511-1908
- *An Introduction to Messaging and Queuing*, GC33-0805
- *MQSeries Intercommunication*, SC33-1872
- *MQSeries Queue Manager Clusters*, SC34-5349
- *MQSeries Clients*, GC33-1632
- *MQSeries System Administration*, SC33-1873
- *MQSeries MQSC Command Reference*, SC33-1369
- *MQSeries Event Monitoring*, SC34-5760
- *MQSeries Programmable System Management*, SC33-1482
- *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390
- *MQSeries Messages*, GC33-1876
- *MQSeries Application Programming Guide*, SC33-0807

- *MQSeries Application Programming Reference*, SC33-1673
- *MQSeries Programming Interfaces Reference Summary*, SX33-6095
- *MQSeries Using C++*, SC33-1877
- *MQSeries Using Java*, SC34-5456
- *MQSeries Application Messaging Interface*, SC34-5604

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, V5.1

> *MQSeries for AIX Quick Beginnings*, GC33-1867

MQSeries for AS/400, V5.1

> *MQSeries for AS/400 Quick Beginnings*, GC34-5557
>
> *MQSeries for AS/400 System Administration*, SC34-5558
>
> *MQSeries for AS/400 Application Programming Reference (ILE RPG)*, SC34-5559

MQSeries for AT&T GIS UNIX V2.2

> *MQSeries for AT&T GIS UNIX System Management Guide*, SC33-1642

MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1

> *MQSeries for Digital OpenVMS System Management Guide*, GC33-1791

MQSeries for Compaq Tru64 UNIX, V5.1

> *MQSeries for Compaq Tru64 UNIX Quick Beginnings*, GC34-5684

MQSeries for HP-UX, V5.1

> *MQSeries for HP-UX Quick Beginnings*, GC33-1869

MQSeries for OS/2 Warp, V5.1

> *MQSeries for OS/2 Warp Quick Beginnings*, GC33-1868

Bibliography

MQSeries for OS/390, V5.2

> *MQSeries for OS/390 Concepts and Planning Guide*, GC34-5650

> *MQSeries for OS/390 System Setup Guide*, SC34-5651

> *MQSeries for OS/390 System Administration Guide*, SC34-5652

> *MQSeries for OS/390 Problem Determination Guide*, GC34-5892

> *MQSeries for OS/390 Messages and Codes*, GC34-5891

> *MQSeries for OS/390 Licensed Program Specifications*, GC34-5893

> *MQSeries for OS/390 Program Directory*

MQSeries link for R/3 Version 1.2

> *MQSeries link for R/3 User's Guide*, GC33-1934

MQSeries for SINIX and DC/OSx, V2.2

> *MQSeries for SINIX and DC/OSx System Management Guide*, GC33-1768

MQSeries for Sun Solaris, V5.1

> *MQSeries for Sun Solaris Quick Beginnings*, GC33-1870

MQSeries for Sun Solaris, Intel Platform Edition, V5.1

> *MQSeries for Sun Solaris, Intel Platform Edition Quick Beginnings*, GC34-5851

MQSeries for Tandem NonStop Kernel, V2.2.0.1

> *MQSeries for Tandem NonStop Kernel System Management Guide*, GC33-1893

MQSeries for VSE/ESA V2.1

> *MQSeries for VSE/ESA Version 2 Release 1 Licensed Program Specifications*, GC34-5365

> *MQSeries for VSE/ESA System Management Guide*, GC34-5364

MQSeries for Windows V2.0

> *MQSeries for Windows User's Guide*, GC33-1822

MQSeries for Windows V2.1

> *MQSeries for Windows User's Guide*, GC33-1965

MQSeries for Windows NT, V5.1

> *MQSeries for Windows NT Quick Beginnings*, GC34-5389

> *MQSeries for Windows NT Using the Component Object Model Interface*, SC34-5387

> *MQSeries LotusScript Extension*, SC34-5404

Softcopy books

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

HTML format

Relevant MQSeries documentation is provided in HTML format with these MQSeries products:
- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1 (compiled HTML)
- MQSeries link for R/3 V1.2

The MQSeries books are also available in HTML format from the MQSeries product family Web site at:

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

Portable Document Format (PDF)

PDF files can be viewed and printed using the Adobe Acrobat Reader.

If you need to obtain the Adobe Acrobat Reader, or would like up-to-date information about the platforms on which the Acrobat Reader is supported, visit the Adobe Systems Inc. Web site at:

```
http://www.adobe.com/
```

PDF versions of relevant MQSeries books are supplied with these MQSeries products:
- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1
- MQSeries link for R/3 V1.2

PDF versions of all current MQSeries books are also available from the MQSeries product family Web site at:

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

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 Version 2 products. Books in PostScript format can be printed on a PostScript printer or viewed with a suitable viewer.

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

The MQSeries product family Web site is at:

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

By following links from this Web site you can:

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

Related publications

This section lists the publications available for related products.

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

Related publications

Index

Numerics

16-bit mode 145
32-bit mode 145

A

access control 117
add routing entry 95
addrtge 95
ADDRTGE 95
advantages of using MQSeries clients 5
AIX client
  components for 31
  custom installation 34
  error messages 158
  hardware and software 14
  installing 33
  removing 37
  trace 161
amqclchl.tab 133
AMQERR01.FDC 133
AMQERR01.LOG 133
AMQSGET0 sample program 111
AMQSGETC sample program 105, 111
AMQSGETW sample program 105
AMQSPUT0 sample program 110
AMQSPUTC sample program 105, 109
AMQSPUTW sample program 105, 110
applications
  building 143
  connected to multiple queue
    managers 151
  connected to multiple servers 151
  connection to server 151
  designing 140
  in different environments 144
  non-version 5 12
  version 5.1 12
applications on MQSeries clients 5
ApplType 144
AS/400
  CL commands 121
AT&T GIS UNIX
  hardware and software 16
  NCR UNIX client 16
authentication 115
automatic definition of channels 120
AUTOSTART 96

B

base product 73
benefits of using MQSeries clients 5
bibliography 183
BookManager 185
building applications 143
building applications for MQSeries clients
  MQI 128

C

C++ applications
  linking 143, 149

C++ applications *(continued)*
  using 143
C applications
  linking 147
CCSID (coded character set
 identifier) 139
CEDA 146
channel
  client connection 120, 152
  definition
    automatic 120
    by user 121
    connecting to a queue-sharing
     group 129
    on the client and server 121
    using MQSC commands 121
  definition of client 122
  message 119
  MQI 119
  overview 119
  server connection 120
  server-connection 120, 124
  setting environment variables 125
  starting 106
  user-defined 121
channel definition
  definition of server 124
  maximum message length 139
  on the client 122
  on the server 121, 124
  overview 119
channel exits
  ClientExitPath 127, 146
  ExitsDefaultPath 127, 146
  overview 127, 146
  path to exits 127, 146
  receive 127, 146
  security 127, 146
  send 127, 146
channel initiator, starting
  OS/390 example 108
CICS applications (non-OS/390)
  CEDA 146
  CICSENV.CMD file 146
  environment specifics 146
  MQSERVER 146
CL commands
  AS/400 121
  OS/400 121
class of routing entry 96
Client CD-ROM 29
client channel definition table 126
  directory path 132
  how it is used 152
  name of 133
  where to find it 133, 151
client connection, defining 125
client-connection channel
  defining 120
  example 108
client files on the server 73

client library file 147
client setup
  example 108
client to server connection 4, 120
ClientExitPath 127, 146
clients (Version 5.1)
  removing 30
clients available from Internet 12
clients overview 3
CLNTCONN 125
COBOL
  link libraries 149
  linking applications 149
coded character set identifier
 (CCSID) 139
commands
  CL 121
  MQSC 121
communication protocol
  DECnet 83
  LU 6.2 83
  NetBIOS 83
  overview 12
  SPX 83
  TCP/IP 83
communication type 83
communications
  configuring 83
  overview 12
Communications Manager/2 92
Compaq Tru64 UNIX client
  hardware and software 17
  installing 38
  NLSPATH environment variable 39
  translated messages 39
  uninstalling 39
components
  OS/2 Warp client 45
  Sun Solaris client 53
  Windows 3.1 client 58
  Windows 95 client 60
  Windows 98 client 60
  Windows NT client 64
configuring communications 83
connection
  client to server 4, 120
  DECnet 102
  LU 6.2 91
  NetBIOS 98
  overview 12
  queue managers 151
  server to client 120
  SPX 100
  TCP/IP 85
  to queue-sharing group 128
creating
  groups
    on AIX client 32
    on Compaq Tru64 UNIX client 38
    Sun Solaris, Intel Platform
     Editionclient 56

187

Sending your comments to IBM

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

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

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

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

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

You can send your comments to IBM in any of the following ways:
- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:
- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

IBM ®