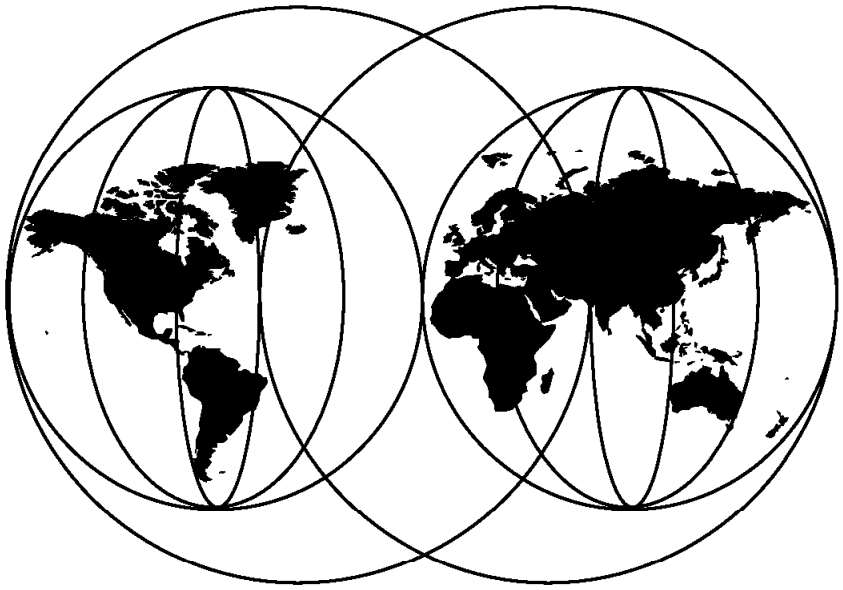
Using MQSeries on the AS/400

Dieter Wackerow, Klaus Haaber-Bernth



International Technical Support Organization

http://www.redbooks.ibm.com

This book was printed at 240 dpi (dots per inch). The final production redbook with the RED cover will be printed at 1200 dpi and will provide superior graphics resolution. Please see "How to Get ITSO Redbooks" at the back of this book for ordering instructions.

SG24-5236-00

SG24-5236-00

International Technical Support Organization

Using MQSeries on the AS/400

July 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 193.

First Edition (July 1998)

This edition applies to MQSeries for AS/400 Version 4 Release 2.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

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

© **Copyright International Business Machines Corporation 1998. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1998

Figures

© Copyright IBM Corp. 1998

Tables

© Copyright IBM Corp. 1998

Preface

This redbook will help you get started with MQSeries for AS/400 V4R2. It explains why and how to use MQSeries with the OS/400 operating system and shows that MQSeries is more than just another AS/400 queuing system. The book is written for two audiences:

- Persons who know the AS/400 but have only a little understanding of MQSeries and how to use it
- Persons who know MQSeries but not the AS/400 and need some extra guidance on how to use OS/400 with MQSeries

This redbook includes an overview of MQSeries explaining its architecture, how to connect clients to servers and servers to servers, how to trigger applications, and the MQI programming interface. Another chapter explains to AS/400 novices how to get started using OS/400. It describes the command interface, how the libraries are used and how to edit, compile and test a program.

The book provides detailed information on installation and migration you cannot find anywhere else. It contains guidelines about what user types require what authority to work with MQSeries objects.

This redbook explains what has to be done to use MQSeries for communication between an AS/400 and other platforms, such as Windows NT. It provides examples that guide you through the administrative process of defining connections between different systems.

It also contains guidelines for daily MQSeries operations, such as journaling, making backups, and defining MQSeries objects to support new applications. Information regarding security, backup, restore and recovery are also included.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Center, Raleigh.

Klaus Haaber-Bernth is a Senior Systems Engineer in the Service Organization in IBM Sweden. He holds a degree in Chemical Engineering from the Technical University in Copenhagen. He has worked on AS/400 and the previous systems S/38, S/36, S/34, S/32 and S/3 in his 30 years with IBM. Klaus has been in the Industry Segment and in AS/400 Marketing. He has conducted courses in Communications, Performance and Systems

© Copyright IBM Corp. 1998

Design. He has also been on a residency in Rochester with Journaling. One of his present tasks in Sweden is services around the Year 2000 challenge for AS/400 customers and Business Partners.

Dieter Wackerow is an Advisory ITSO Specialist for MQSeries at the Systems Management and Networking ITSO Center, Raleigh. His areas of expertise include application design and development for various industries, performance evaluations, capacity planning, and modelling of computer systems and networks. He also wrote a simulator for banking hardware and software. He taught classes and has written on performance issues, application development and about MQSeries.

Thanks to the following people for their invaluable contributions to this project:

Gottfried Schimunek
International Technical Support Organization, Rochester Center

Hubert T. Lye
IBM Australia

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 207 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM Intranet users `http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. Why Should You Use MQSeries on an AS/400?

The AS/400 operating system, OS/400, has already two queue systems, message queues (MSGQ) and data queues (DTAQ). Message queues are used for system and program messages for the operator. Data queues are used to exchange data between applications in one or more AS/400 systems, and between an AS/400 and clients (PCs) using the Client Access for AS/400 software. So, why should you use another queuing system, MQSeries, on an AS/400?

1.1 Is MQSeries Just Another AS/400 Queuing System?

MQSeries means commercial messaging and queuing. It allows business applications to integrate and communicate across a spectrum of desktop and mainframe systems, overcoming inconsistencies with different network protocols and all major commercial platforms. Key benefits of this award-winning market leader are:

- A single, multi-platform API
- Isolation from communications programming
- Time-independent processing
- Assured message delivery, even when programs or networks fail
- Robust middleware for high-performance distributed application integration

MQSeries lets you break away from the constraints of differing technologies while maintaining the benefits. It provides a scalable infrastructure integrating distributed applications.

The key advantages of MQSeries over AS/400 data queuing are:

- MQSeries message queues are secure. Persistent messages survive a system restart.
- MQSeries enables application programs running on different platforms to communicate with each other using the Message Queuing Interface (MQI) which is consistent over all platforms.

If the following issues are important to you, MQSeries for AS/400 Version 4 Release 2 is for you:

- MQSeries is a product family for more than 25 different platforms. Data queues and message queues are only for AS/400 and ClientAccess/400 PCs.

© Copyright IBM Corp. 1998

- Design, programming and administration is similar on different MQSeries platforms.
- An application written to MQSeries standards is more likely to be portable to other platforms.
- MQSeries for AS/400 can store queue data persistently. AS/400 standard data queues cannot.
- MQSeries for AS/400 can synchronize transactions and AS/400 databases with commit/rollback. AS/400 standard data queues cannot.
- Channel sequence checking between systems assures that no data is lost (not implemented for AS/400 standard remote data queues).
- MQSeries allows remote administration across platforms.
- MQSeries for AS/400 uses both the standard MQSeries commands and the OS/400 command interface, well known to AS/400 people.
- MQSeries isolates the communication interface as easy as AS/400 standard data queue does.
- MQSeries for AS/400 can browse a message (non-destructive read). AS/400 data queues cannot.
- Version 4 Release 2 of MQSeries for AS/400 uses the new *heartbeat* function to keep channel management from waiting for nothing.
- Version 4 Release 2 of MQSeries for AS/400 allows distribution lists to minimize unnecessary data flow. Version 4 Release 2 of MQSeries for AS/400 supports blocking and deblocking. This eliminates the need for application blocking and deblocking, and thus makes it straightforward to use commit/rollback for synchronizing queues and databases.

1.2 About MQSeries for AS/400 Version 4 Release 2

MQSeries for AS/400 Version 4 Release 2 is a new product, 5769-MQ1, with new terms and conditions. A migration path is possible for these previous versions of MQSeries for AS/400:

- 5733-103 V2: for OS/400 V2R3
- 5763-MQ1 V3R0.5 and V3R1: for OS/400 V3R0.5 and V3R1
- 5763-MQ2 V3R2: for OS/400 V3R2
- 5716-MQ1 V3R6: for OS/400 V3R7 and V4R1

V4.2 has a user-based price structure. A user is defined as a concurrent MQSeries connection. An OS/400 job running an MQSeries application counts as a user from the execution of MQCONN until MQDISC. Also an MQSeries administration task counts as a user, when running.

More information about this can be obtained from your IBM re-marketer/distributor or IBM representative.

1.3 MQSeries V5 and MQSeries for AS/400 V4R2

MQSeries for AS/400 V4R2 has implemented some, but not all features of MQSeries Version 5.

The following URL gets you to the Web page that describes the new features of *MQSeries Version 5*:

`http://www.software.ibm.com/ts/mqseries/v5`

Some of the features are:

- Database coordination (MQSeries as a resource manager)
- Smart distribution for multicast messages (dynamic distribution lists with late fan out)
- Performance boosts for distributed messages
- New programming features for support of very large messages and files (message segmenting and grouping, and reference messages)
- Novell SPX support
- Enhancements in administration, problem resolution, and security

The IBM announcement letter for *MQSeries for AS/400 Version 4 Release 2* says:

In this release, MQSeries also adds important new features to exploit family synergy:

- Simplified application design and improved performance through distribution lists which allow a single message to be put to multiple queues
- Improved programmer choice through automatic creation of channel definitions for receiver and server-connection channels
- Static bindings for the ILE RPG programming language and support for Message Queuing Interface (MQI) applications written in C++ increase programmer choice
- Message segmentation, ordering and grouping to improve checking of transactional data and allow more applications to use MQSeries for AS/400, particularly for large transactions
- Fast nonpersistent messages for data which needs simple, fast delivery

- Channel heartbeats to provide faster response when the system is stopping or resetting

1.4 What Is Familiar in MQSeries to the AS/400 Professional?

Queuing is not new to the AS/400, neither is hiding complexity. MQSeries is much the same across platforms. Nevertheless, it is well integrated in the AS/400 and as user friendly as any other AS/400 product.

1.4.1 Asynchronous Application Design

Both MQSeries queues and AS/400 data queues make asynchronous application design possible. You have the same freedom to design your own message formats, and you have the same simple interface. MQSeries′ concept of channels, though, is new.

1.4.2 Message and Message Descriptor

Messages consist of data and information about the data. When you receive data from an AS/400 message queue or data queue, you can specify that you want information about the sender, the message ID, CCSID (character set), and more.

MQSeries messages do not only contain the message data, but also information about the message in the *Message descriptor* (message header), such as: message ID, correlation ID, user ID, CCSID, format name, reply-to-queue, priority and much more. The MQSeries queue manager uses the header information. Some of the fields can be modified and used by the application programmer.

1.4.3 Retrieving Messages from Queues

You can read messages from queues sequentially or you may select a specific message. In MQSeries, you provide a *message ID* and/or a *correlation ID*. In the AS/400, the *key field* is used to specify that you are only interested in certain types of messages.

Usually, messages are retrieved first-in-first-out. However, it is possible to specify a priority and also an expiration date.

1.4.4 Remote Data Queues

Remote data queues point to queues on a remote system. AS/400 data queues can be created to point at a queue in a remote system (DDM distributed data management). An MQSeries remote queue definition also points to a queue in a remote system. With MQSeries, you don′t have to specify any communication information in the remote queue definition. This is done in the channel definitions.

1.4.5 MQSeries Queues Are Secure

MQSeries messages are secured like data in an AS/400 file. Data in an MQSeries queue (MQMQ) is stored in an AS/400 object type (dataspace), and AS/400 journaling is used to secure the data. The journal management is standard AS/400 Journal Management.

1.4.6 MQSeries Clients and ClientAccess/400

MQSeries clients work like ClientAccess/400 clients. MQSeries clients run on a variety of platforms, such as Windows 95, Windows 3.1, OS/2, and AIX. They must be connected to an MQSeries server.

Clients do have queues of their own; however, those queues reside in the server. MQSeries client applications use APIs in the same way as any PC program may use the programming APIs supplied by ClientAccess/400 in order to access a data queue on AS/400.

Users in a mobile environment use a different product, MQSeries for Windows Version 2.1. These clients, also called leaf nodes, have a queue manager and queues of their own. MQSeries for AS/400 will treat mobile users as any other remote MQSeries queue manager (server system). Mobile users can work even when they are disconnected, just as mobile Lotus Notes users can work in island mode.

1.4.7 MQSeries Objects Are AS/400 Objects

In an AS/400, users are referred to as user profiles. MQSeries for AS/400 commands are AS/400 commands. Normal access security is used. MQSeries queues, channels, and other objects are stored in the AS/400 object type dataspace. This has been done to prevent inventing new object types in the AS/400.

MQSeries naming conventions allow names to be 48 characters long while AS/400 names consist of a 10-character object name combined with a 10-character library name. You can see the mapping of the MQSeries names to AS/400 names by executing the MQM/400 Display MQM Object Names command:

```
/DSPMQMOBJN OBJ(*ALL)
```

MQSeries for AS/400 object names are not known to the OS/400 operating system. Therefore, MQSeries for AS/400 does not use the AS/400 object security (GRTOBJAUT, RVKOBJAUT, etc.), but uses commands of its own, such as: GRTMQMAUT and RVKMQMAUT.

1.4.8 MQSeries Queue Manager

MQSeries for AS/400 allows only one queue manager instance. The AS/400 database also has only one instance. This has not been considered a shortcoming for AS/400. The fact that MQSeries for AS/400 has only one instance should not prohibit multiple usage.

We recommend that you use a prefix in the names to keep different usages separate.

You may name the AS/400 database with the Change RDB Directory Entry command:

```
CHGRDBDIRE RDB(DBSYSA) RMTLOCNAME(*LOCAL)
```

The MQSeries queue manager is named in the Create Message Queue Manager command:

```
CRTMQM MQMNAME(MQSYSA)
```

Note: We recommend that you use the *system name* as queue manager name.

Chapter 2. MQSeries Overview

MQSeries is IBM's award winning middleware for commercial messaging and queuing. It runs on a variety of platforms. The MQSeries products enable programs to communicate with each other across a network of unlike components, such as processors, subsystems, operating systems and communication protocols. MQSeries programs use a consistent application program interface (API) across all platforms.

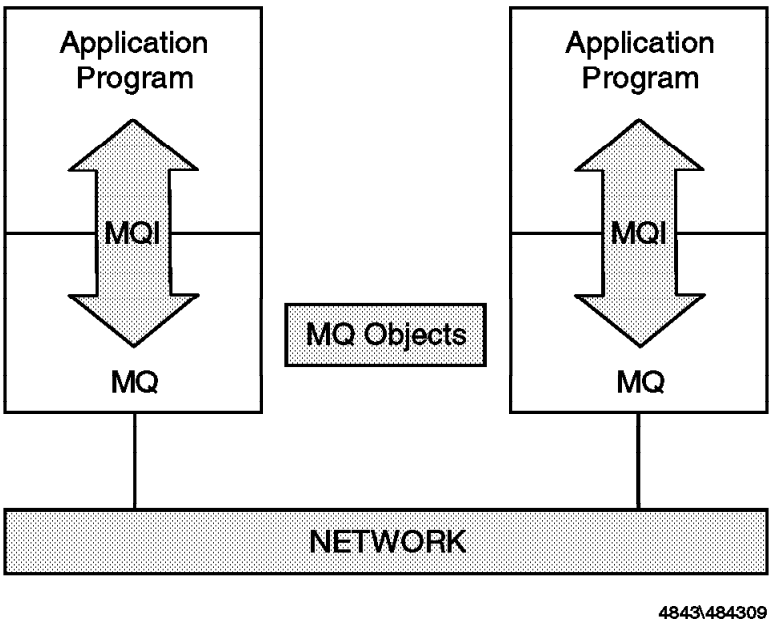


Figure 1. MQSeries at Runtime

Figure 1 shows the parts of an MQSeries application at runtime. Programs use MQSeries API calls, that is the message queue interface (MQI), to communicate with a queue manager (MQM), the runtime program of MQSeries. For the queue manager to do its work, it refers to objects, such as queues and channels. The queue manager itself is an object as well.

The following sections provide a brief overview of MQSeries, including clients and servers.

This chapter discusses issues valid for the AS/400 and other platforms. After all, most MQSeries for AS/400 installations communicate with

© Copyright IBM Corp. 1998

MQSeries on other systems. So the specialist needs to know both common MQ commands and the special AS/400 MQ CL commands.

2.1 What Is Messaging and Queuing?

Message queuing is a method of program-to-program communication. Programs within an application communicate by writing and retrieving application-specific data (messages) to/from queues, without having a private, dedicated, logical connection to link them.

Messaging means that programs communicate with each other by sending data in messages and not by calling each other directly.

Queuing means that programs communicate through queues. Programs communicating through queues need not be executed concurrently.

With asynchronous messaging, the sending program proceeds with its own processing without waiting for a reply to its message. In contrast, synchronous messaging waits for the reply before it resumes processing.

MQSeries is used in a client/server or distributed environment. Programs belonging to an application can run in one workstation or in different machines on different platforms.

2.1.1 Messages

A message consists of two parts: data that is sent from one program to another and a message descriptor. The message descriptor identifies the message (message ID) and contains control information, also called attributes, such as message type, expiry time, correlation ID, priority, and the name of the queue for the reply. In MQSeries Version 5 the maximum message length is 100 MB, but for all other platforms it is 4 MB. Messages can be segmented and grouped. MQSeries knows four types of messages:

Datagram A message containing information for which no response is expected.

Request A message for which a reply is requested.

Reply A reply to a request message.

Report A message that describes an event such as the occurrence of an error.

Application design determines whether a message must reach its destination under any circumstances, or if it can be discarded when it cannot get there in time. MQSeries differentiates between persistent and non-persistent messages. Delivery of *persistent messages* is assured. They

are written to logs and survive system failures. In an AS/400 these logs are Journal Receivers. *Non-persistent* messages cannot be recovered after a system restart.

2.1.2 Queue Manager

The heart of MQSeries is the *message queue manager* (MQM). Its job is to manage queues of messages. Application programs invoke functions of the queue manager by issuing API calls. For example, the MQPUT API puts a message on a queue to be read by another program using the MQGET API. This scenario is shown in Figure 2.

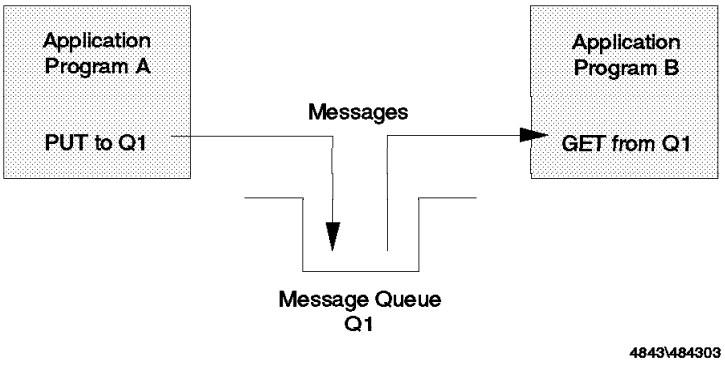


Figure 2. Program-to-Program Communication - One System

A program may send messages to another program that runs in the same machine as the queue manager, or to a program that runs in a remote system, such as a server or a host. The remote system has its own queue manager with its own queues. This scenario is shown in Figure 3.

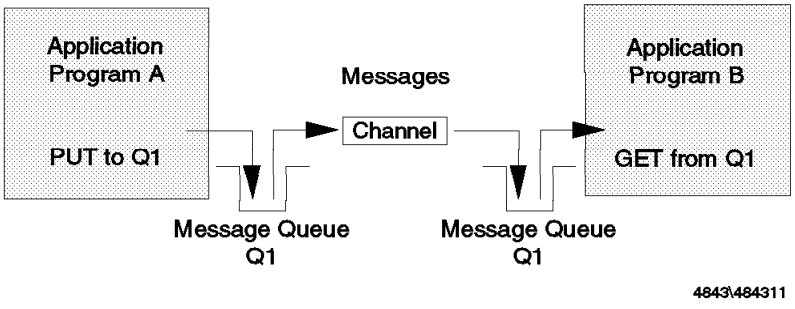


Figure 3. Program-to-Program Communication - Two Systems

Application programmers do not need to know where the program to which they are sending messages runs. They put their message on a queue and let the queue manager worry about the destination machine and how to get it there.

For the queue manager to do its work, it refers to objects that are defined by an *administrator*, usually when the queue manager is created or when a new application is added. The objects are described in the next section.

The functions of a queue manager can be defined as follows:

- It manages queues of messages for application programs.
- It provides an application programming interface, the Message Queue Interface (MQI).

  Note: The Networking Blueprint identifies three communication styles:

  - Common Programming Interface - Communications (CPI-C)
  - Remote Procedure Call (RPC)
  - Message Queue Interface (MQI)
- It uses networking facilities to transfer messages to another queue manager when necessary.
- It provides additional functions that allow administrators to create and delete queues, alter the properties of existing queues, and control the operation of the queue manager. In UNIX, OS/2 and Windows NT, these functions are invoked through the utility RUNMQSC, which stands for run MQSeries commands. In an AS/400, MQSeries commands are executed from a command file via the AS/400 command STRMQMMQSC or interactively via special AS/400 MQ CL commands. You access these functions through the MQSeries Command panel. Type `go cmdmqm` on the command line.

MQSeries clients do not have a queue manager of their own. Client machines connect to a queue manager in a server. This MQM manages the queues for all clients attached to it.

In contrast to MQSeries clients, each workstation that runs MQSeries for Windows has its own queue manager and queues. MQSeries for Windows has a single-user queue manager and is not intended to function as a queue manager for other MQSeries clients.

2.1.3 MQSeries Objects

This section introduces you to queue manager objects, such as queues and channels. The queue manager itself is an object, too. Usually, an administrator creates one or more queue managers and their objects. A queue manager can own objects of the following types:

- Queues
- Process definitions
- Channels

The objects are common across different MQSeries platforms. There are other objects that apply to MVS systems only, such as the buffer pool, PSID, and storage class. AS/400 MQ objects are known to the OS/400 operating system as object type *USRSPC (user space) in the QMQMDATA library.

2.1.3.1 Queues

Message queues are used to store messages sent by programs. There are local queues that are owned by the local queue manager, and remote queues that belong to a different queue manager. Queues are described in more detail in 2.2, “Message Queues” on page 12.

2.1.3.2 Channels

Messages are transmitted via channels. A channel is a logical communication link. In MQSeries, there are two different kinds of channels:

- Message channel

  A message channel connects two queue managers via message channel agents (MQA). Such a channel is unidirectional. It comprises two message channel agents (MCA), a sender and a receiver, and a communication protocol. An MCA is a program that transfers messages from a transmission queue to a communication link or vice versa. For bidirectional messaging you have to define two channels, a sender channel and a receiver channel.

- MQI channel

  A Message Queue Interface (MQI) channel connects an MQI client to a queue manager in a server machine. MQI clients don′t have a queue manager of their own. An MQI channel is bidirectional.

Figure 8 on page 20 shows the use of both channel types. For more detailed information refer to *MQSeries Intercommunication*, SC33-1872.

To transmit non-persistent messages, a message channel can run with at speeds: fast and normal. Fast channels improve performance, but messages are lost if there is a channel failure.

A channel can use one of the following transport types:

- SNA LU 6.2
- TCP/IP
- NetBIOS (not AS/400)
- SPX (not AS/400)

Note: MQSeries for Windows Version 2.1 uses message channels to connect to other machines. Since this product is designed as a single user system, it does not support MQI channels. MQSeries for Windows supports only TCP/IP.

2.1.3.3 Process Definitions

A process definition object defines an application to a queue manager. For example, it contains the name of the program to trigger when a message arrives.

Notes:

1. For the AS/400, only process definitions to trigger applications are needed.
2. With MQSeries Version 5, process definitions for channels have been eliminated.

2.2 Message Queues

Queues are defined as objects belonging to a queue manager. MQSeries knows a number of different queue types, each with a specific purpose.

2.2.1 Local Queue

A queue is local if it is owned by the queue manager to which the application program is connected. They are used to store messages for programs that use the same queue manager. For example, program A and program B each has a queue for incoming messages and another queue for outgoing messages. Since the queue manager serves both programs, all four queues are local.

Note: Both programs do not have to run in the same workstation. Client workstations usually use a queue manager in a server machine.

2.2.2 Remote Queue

A queue is remote if it is owned by a different queue manager. It is the *local definition* of a remote queue. A remote queue definition is associated with a transmission queue.

Applications do not need to know the location of the remote queue. Programs write messages to queues. The local queue manager is responsible for forwarding the messages to the remote queue manager.

Note: A program cannot read messages from a remote queue.

2.2.3 Transmission Queue

A remote queue is associated with a transmission queue. Transmission queues are local and used as an intermediate step when sending messages to remote queues.

Typically, there is only one transmission queue for each remote queue manager (or machine). All messages written to queues owned by a remote queue manager are actually written to the transmission queue for this remote queue manager. The messages will then be read from the transmission queue and sent to the remote queue manager.

Transmission queues are transparent to the application. They are used internally by the queue manager.

Note: When a program opens a remote queue, the attributes of the queue are obtained from the transmission queue. Therefore, the results of a program writing messages to a queue will be affected by the transmission queue characteristics.

2.2.4 Dynamic Queue

Such a queue is defined ″on the fly″ when the application needs it. Dynamic queues may be retained by the queue manager or automatically deleted when the application program ends.

Dynamic queues are local queues. They are often used in conversational applications, to store intermediate results. Dynamic queues can be:

- Temporary queues that do not survive queue manager restarts
- Permanent queues that do survive queue manager restarts

2.2.5 Model Queue

A model queue is not a real queue. It is a collection of attributes that are used when a dynamic queue is created.

2.2.6 Alias Queue

Alias queues are not real queues but definitions. They are used to assign different names to the same physical queue. This allows multiple programs to work with the same queue, accessing it under different names and with different definitions.

2.2.7 Initiation Queue

An initiation queue is a local queue to which the queue manager writes a *trigger message* when certain conditions are met on another local queue, for example, when a message is put into an empty message queue. Trigger messages are read by the trigger monitor, an MQSeries application. The trigger monitor then starts the application that will process the message.

Note: Applications do not need to be aware of initiation queues, but the triggering mechanism implemented through them is a powerful tool to design and write asynchronous applications.

2.2.8 Reply-To Queue

A request message must contain the name of the queue into which the responding program must put the reply message.

2.2.9 Dead-Letter Queue

A queue manager must be able to handle situations when it cannot deliver a message. Here are some examples:

- The destination queue is full.
- The destination queue does not exist.
- Message puts have been inhibited on the destination queue.
- The sender is not authorized to use the destination queue.
- The message is too large.
- The message contains a duplicate message sequence number.

These messages are written by the queue manager to a dead-letter queue. A dead-letter queue is defined when the queue manager is created. It will be used as a repository for all messages that cannot be delivered.

Note: From the sender's point of view, only the remote queue manager can put messages into a dead-letter queue. Messages that cannot be delivered by the local queue manager (maybe, because the queue is not defined) cause an error message.

2.3 Creating a Queue Manager

Except for the AS/400, you may create as many queue managers as you like and have them running at the same time. The AS/400 allows only one queue manager instance. It is created via the CRTMQM panel. On platforms other than the AS/400, one of them should be the default queue manager. Create a queue manager with the command `crtmqm`; to make it

the default, specify the parameter /q. The following command creates the default queue manager MYQMGR (in an NT environment):

```
crtmqm /q MYQMGR
```

Note: Queue manager names are case-sensitive.

In AS/400 we would use the MQ CL Command:

```
CRTMQM MQMNAME(MYQMGR)
```

There are some default definitions for objects every queue manager needs. In MQSeries Version 5, 22 objects are created automatically. The file AMQSCOMA.TST, supplied with prior versions, does not exist any longer. Most certainly, you will have to create other objects that pertain to the applications you run.

On non-Version 5 platforms (and AS/400 Version 4 Release 2) you still have to create the default objects with the command (NT environment):

```
runmqsc < c:\mqm\mqsc\amqscoma.tst > out.lst
```

To create the default objects on the AS/400 type:

```
call qmqm/amqsdef4
```

Note: A detailed description on how to create the queue manager and its default objects in an AS/400 environment can be found in 4.5, “Creating the Queue Manager” on page 93.

AS/400 also has a AMQSCOMA file (same definitions as AMQSDEF4). The commands are executed with the following AS/400 MQ commands:

```
STRMQMMQSC SRCMBR(AMQSCOMA) SRCFILE(QMQMSAMP/QMQSC) OPTION(*VERIFY)
STRMQMMQSC SRCMBR(AMQSCOMA) SRCFILE(QMQMSAMP/QMQSC) OPTION(*RUN)
```

The following screen shows how to create a default queue manager with the name MQNT1. The commands you type are shown in bold.

```
C:\crtmqm /q MQNT1
MQSeries queue manager created.
Creating or replacing default objects for MQNT1.
Default objects statistics : 22 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

For each queue manager in NT, a directory is created, such as:

```
C:\MQM\qmgrs\MQNT1
```

The directory contains, among other things, the default objects and the qm.ini file.
The objects automatically created for you (in Version 5) are:

- Queues:
  - SYSTEM.ADMIN.CHANNEL.EVENT
  - SYSTEM.ADMIN.COMMAND.QUEUE
  - SYSTEM.ADMIN.PERFM.EVENT
  - SYSTEM.ADMIN.QMGR.EVENT
  - SYSTEM.CHANNEL.INITQ
  - SYSTEM.CICS.INITIATION.QUEUE
  - SYSTEM.DEAD.LETTER.QUEUE
  - SYSTEM.DEFAULT.ALIAS.QUEUE
  - SYSTEM.DEFAULT.INITIATION.QUEUE
  - SYSTEM.DEFAULT.LOCAL.QUEUE
  - SYSTEM.DEFAULT.MODEL.QUEUE
  - SYSTEM.DEFAULT.REMOTE.QUEUE
  - SYSTEM.MQSC.REPLY.QUEUE
- Process:
  - SYSTEM.DEFAULT.PROCESS
- Channels:
  - SYSTEM.AUTO.RECEIVER
  - SYSTEM.AUTO.SVRCONN
  - SYSTEM.DEF.RECEIVER
  - SYSTEM.DEF.REQUESTER
  - SYSTEM.DEF.SENDER
  - SYSTEM.DEF.SERVER
  - SYSTEM.DEF.SVRCONN

A dead-letter queue is not automatically created. To create one when you create the queue manager, specify it as shown in the following example:

```
crtmqm /q /u system.dead.letter.queue MQNT1
```

To start the default queue manager issue the command: `strmqm`.

For the AS/400, you may also specify the dead-letter queue when you create the queue manager:

```
CRTMQM MQMNAME(MYMQMGR) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

Alternativly, you may add it afterwards:

```
CHGMQM MQMNAME(MYMQMGR) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

STRMQM also starts the MQ Manager in the AS/400.

2.4 Manipulating MQM Objects (RUNMQSC)

MQSeries for systems other than the AS/400 provides the utility RUNMQSC to create and delete queue manager objects and to manipulate them. The queue manager must be running when you use the utility. RUNMQSC works in two ways:

- You can type the commands.
- You can create a file containing a list of commands and use this file as input.

The AS/400 cannot run the MQSC commands interactively. The AS/400 can execute MQSC commands from a batch file using the command:

```
STRMQMMQSC
```

Interactively, the AS/400 uses its own MQ CL commands. A menu for all MQ CL commands is displayed by entering GO CMDMQM on an AS/400 command line.

```
[C:\]strmqm
MQSeries queue manager running.

[C:\]runmqsc
84H2001,6539-B42 (C) Copyright IBM Corp. 1994, 1997. ALL RIGHTS RESERVED
Starting MQSeries Commands.

define qlocal('QUEUE1') replace descr ('test queue')
     1 : define qlocal('QUEUE1') replace descr ('test queue')
AMQ8006: MQSeries queue created.
alter qmgr deadq(system.dead.letter.queue)
     2 : alter qmgr deadq(system.dead.letter.queue)
AMQ8005: MQSeries queue manager changed.
end
     3 : end
2 MQSC commands read.
0 commands have a syntax error.
0 commands cannot be processed.

[C:\]
```

Figure 4. RUNMQSC - Interactive

The commands in Figure 4 start the default queue manager and create the local queue QUEUE1 for it. Another command alters the queue manager properties by defining a dead-letter queue.

To start the utility in an interactive mode, type runmqsc. To end it, type end. This command is new in Version 5, however, Ctrl+C still works.

Another way to create MQSeries objects is by using an input file instead of typing the commands. Figure 5 shows the command that processes the input file in Figure 6. The response from RUNMQSC (file a.a) is shown in Figure 7 on page 19.

```
[C:\]strmqm
MQSeries queue manager running.

[C:\]runmqsc < mycoma.tst > a.a

[C:\]
```

Figure 5. RUNMQSC - Using Command File

The input file mycoma.tst contains the lines shown in Figure 6. The + indicates that the command continues on the next line.

```
****************************************************
* File: MYCOMA.TST
****************************************************

    DEFINE QLOCAL('QUEUE1') REPLACE +
           DESCR('Test Queue')
```

Figure 6. RUNMQSC - Input File, or AS/400 STRMQMMQSC

The output can appear in the window or can be redirected to a file by specifying a > followed by a file name. The output for the above file would look as shown in Figure 7 on page 19.

The AS/400 can execute the same MQSC commands from a file member in a source file. Use the following command:

```
STRMQMMQSC SRCMBR(MYCOMA) SRCFILE(MYLIB/TST)
```

The command file is contained in the member MYCOMA in the source file TST in the library MYLIB. The command has an option for verifying or running. The AS/400 writes the result to a spool file.

```
5684H2001,6539-B42 (C) Copyright IBM Corp. 1994, 1997.  ...
Starting MQSeries Commands.

       : ************************************************
       : * File: MYCOMA.TST
       : ************************************************
       :
     1 : DEFINE QLOCAL ('QUEUE1') REPLACE +
       :        DESCR ('Test Queue')
AMQ8006: MQSeries queue created.
       :
1 MQSC commands read.
0 commands have a syntax error.
```

Figure 7. RUNMQSC - Output File, or AS/400 WRKSPLF

2.5 Clients and Servers

MQSeries distinguishes clients and servers. On AS/400 you can only install the server. Before you install MQSeries on NT and AIX, you have to decide if the workstation will be an MQSeries client, an MQSeries server, or both. With MQSeries for Windows a new term was introduced, the leaf node (described on page 20). There are two kinds of clients:

- Slim client or MQSeries client
- Fat client

Fat clients have a local queue manager; slim clients don't.

When a slim client cannot connect to its server it cannot work, because the queue manager and queues for a slim client reside in the server. Usually, an MQSeries client is a slim client. Several of these clients share MQSeries objects, and the queue manager is one of them, in the server they are attached to.

Note: The MQSeries Client for Java is a slim client.

In some cases it may be advantageous to have queues in the end user's workstation, especially in a mobile environment. That allows you to run your application when a connection between client and server does not (temporarily) exist.

You may install client and server software in the same system and use it as an end user's workstation. If your operating system is Windows NT you can install MQSeries for Windows NT V5.0 or MQSeries for Windows V2.1

(MQWin). If your operating system is Windows 95 use MQWin V2.1. This product has been designed for end users and uses fewer resources.

The difference between an end user's workstation that is a client and one that has a queue manager is the way messages are sent. The queues reside either in the end user's workstation or in the server.

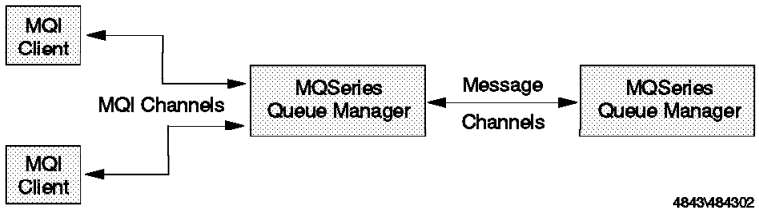


Figure 8. MQSeries Channels

Figure 8 shows the use of MQI and message channels.

- MQI channels connect clients to a queue manager in a server machine. All MQSeries objects for the client reside in the server. MQI channels are faster than message channels.
- A message channel connects a queue manager to another queue manager in another system.

The following summarizes the three workstation types:

MQSeries client
: A client workstation does not have a queue manager of its own. It shares a queue manager in a server with other clients. All MQSeries objects, such as queues, are in the server. Since the connection between client and server is synchronous, the application cannot work when the communication is broken. One could refer to such workstations as "slim" clients.

MQSeries server
: A workstation can be a client and a server. A server is an intermediate node between other nodes. It serves clients that have no queue manager and manages the message flow between its clients, itself and other servers.

  In addition to the server software you may install the client software, too. This configuration is used in an application development environment.

Leaf node MQSeries for Windows was designed for use by a single user. It has its own "small footprint" queue manager with its own objects. However, it is not an intermediate node between other nodes. It

is called a leaf node. One could also refer to it as a ″fat″ client. This product is able to queue outbound messages when connection to a server or host is not available, and inbound messages when the appropriate application is not active.

2.6 How MQSeries Works

Figure 9 on page 22 shows the parts and architecture of MQSeries. The application program uses the Message Queue Interface (MQI) to communicate with the queue manager. The MQI is described in more detail in 2.9, “The Message Queuing Interface (MQI)” on page 39. The queuing system consists of the following parts:

- Queue Manager (MQM)
- Listener
- Trigger Monitor
- Channel Initiator
- Message Channel Agent (MCA)

When the application program wants to put a message on a queue it issues an MQPUT API. This invokes the MQI. The queue manager checks if the queue referenced in the MQPUT is local or remote. If it is a remote queue, the message is placed into the transmission (xmit) queue. The queue manager adds a header that contains information from the remote queue definition, such as destination queue manager name and destination queue name.

Each remote queue must be associated with an xmit queue. Usually, all messages destined for one remote machine use the same xmit queue.

Transmission is done via channels. Channels can be started manually or automatically. To start a channel automatically, the xmit queue must be associated with a channel initiation queue. Figure 9 on page 22 shows that the queue manager puts a message into the xmit queue and another message into the channel initiation queue. This queue is monitored by the *channel initiator*.

The channel initiator is an MQSeries program that must be running in order to monitor initiation queues. When the channel initiator detects a message in the initiation queue, it starts the message channel agent (MCA) for the particular channel. This program moves the message over the network to the other machine, using the sender part of the uni-directional message channel pair.

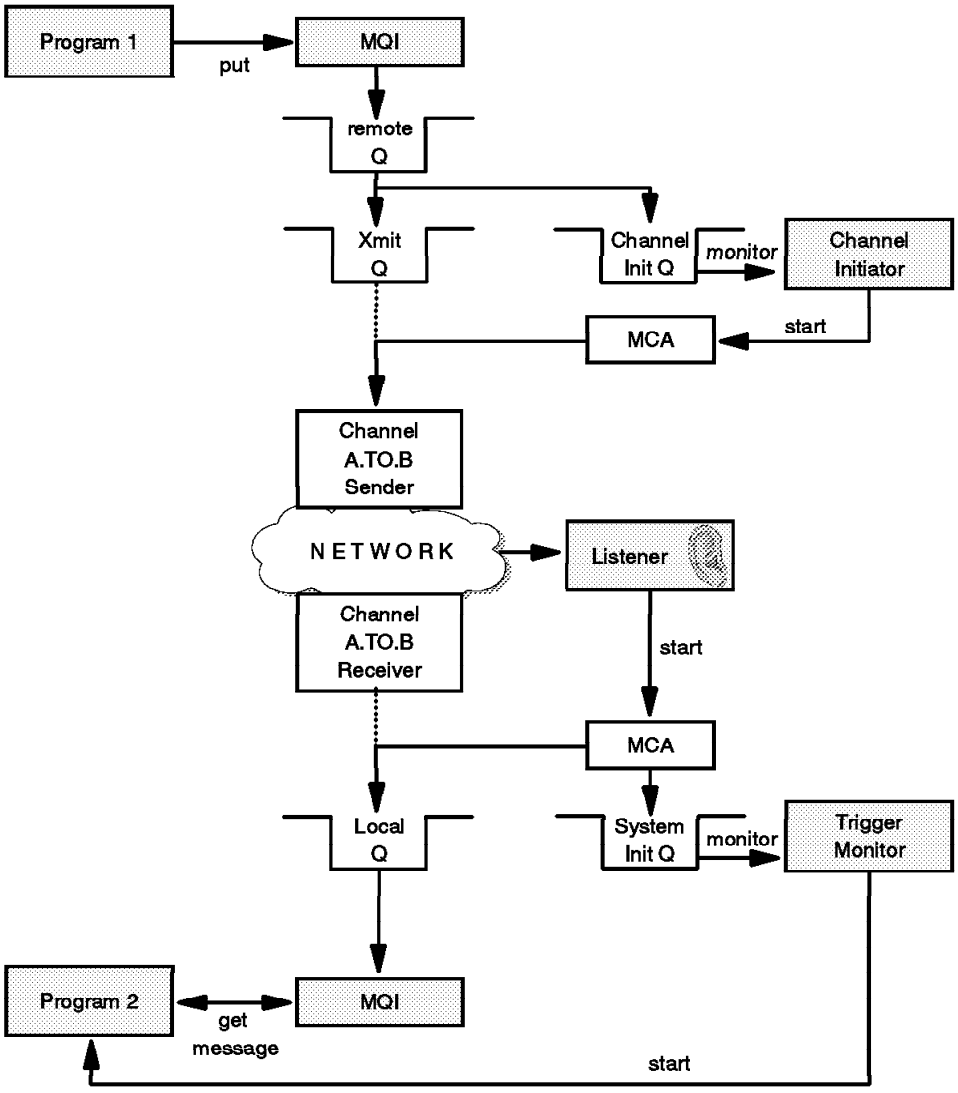


Figure 9. MQSeries Parts and Logic

On the receiving end, a *listener* program must have been started. The listener, also supplied with MQSeries, monitors a specified port, by default the port dedicated to MQSeries, 1414. When a message arrives, it starts the *message channel agent*. The MCA moves the message into the specified local queue using the receiver part of the message channel pair.

Note: Both channel definitions, sender and receiver, must have the same name. For the reply, you need another message channel pair.

The program that processes the incoming message can be started manually or automatically. To start the program automatically, an initiation queue and a process must be associated with the local queue, and the *trigger monitor* must be running.

When the program shall start automatically, the MCA puts the incoming message into the local queue and a trigger message into the initiation queue. This queue is monitored by the trigger monitor. This program invokes the application program specified in the process definition. The application issues an MQGET API to retrieve the message from the local queue.

This scenario is described in more detail in the next section.

2.7 Communication between Queue Managers

In this section, we discuss what you have to define to send messages to a queue manager that resides in another system. We use message channels for communication between queue managers (see Figure 8 on page 20).

Each machine has a queue manager installed and each queue manager manages several local queues. Messages destined for a remote queue manager are put into a *remote queue*. A remote queue is not a real queue; it is the definition of a local queue in the remote machine. A remote queue is associated with a transmission (xmit) queue which is a local queue. Usually, there is one xmit queue for each remote queue manager.

A transmission queue is associated with a message channel. Message channels are uni-directional, meaning that you have to define two channels for a conversational type of communication. Also, you have to define each channel twice, once in the system that sends the message (sender channel) and once in the system that receives the message (receiver channel). Each channel pair (sender and receiver) must have the same name.

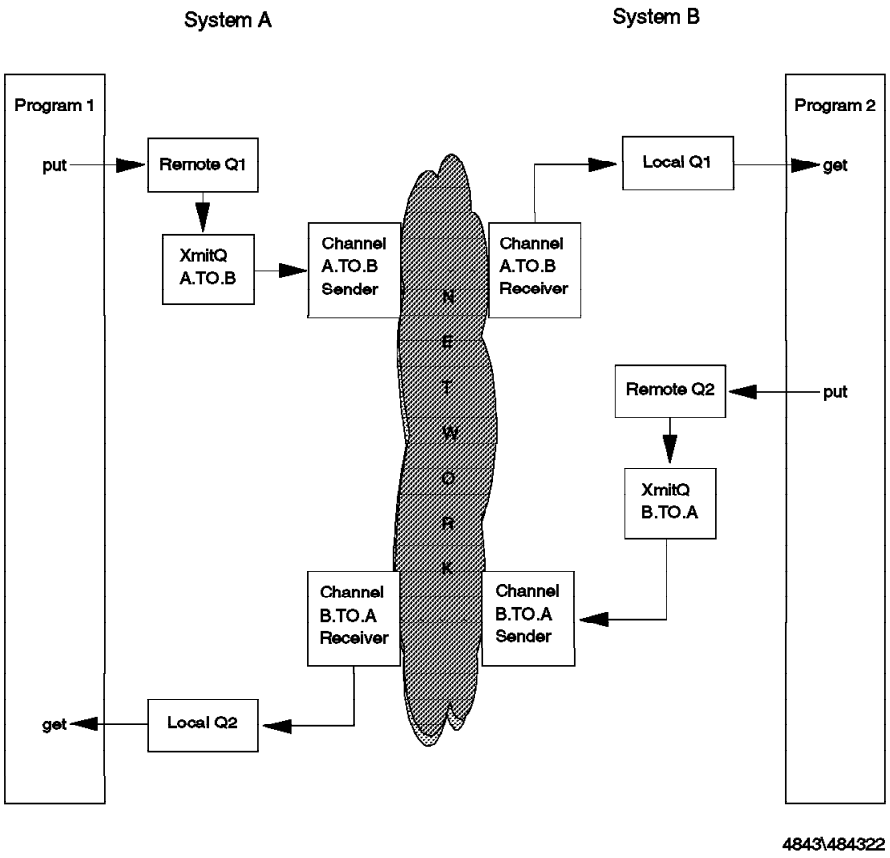


Figure 10. Communication between Two Queue Managers

2.7.1 How to Define a Connection between Two Systems

Figure 10 shows the required MQSeries objects for connecting two queue managers. In each system we need:

- A remote queue definition that mirrors the local queue in the receiver machine and links to a transmission queue (Q1 in system A and Q2 in system B)
- A transmission queue that holds all messages destined for the remote system until the channel transmits them (B in system A and A in system B)
- A sender channel that gets messages from the xmit queue and transmits them to the other system using the existing network (A.TO.B in system A and B.TO.A in system B)

Table 1. MQSeries Objects for MQM-to-MQM Connection

Sender (System A)	Receiver (System B)
DEFINE QREMOTE('Queue1') REPLACE + RNAME('Queue1') + RQMNAME(SYSTEMB) + XMITQ(B) + DESCR('Queue 1 in system B') DEFINE QLOCAL(SYSTEMB) REPLACE + USAGE(xmitq) + DESCR('Xmit Queue') DEFINE CHANNEL(A.TO.B) + CHLTYPE(sdr) REPLACE + TRPTYPE(tcp) CONNAME(9.24.104.116) + XMITQ(SYSTEMB) + DESCR('Sender channel from A to B') DEFINE QLOCAL('Queue2') REPLACE + DESCR('Messages from system B') DEFINE CHANNEL(B.TO.A) + CHLTYPE(rcvr) REPLACE + TRPTYPE(tcp) + DESCR('Receiver channel from B to A')	DEFINE QLOCAL('Queue1') REPLACE + DESCR('Messages from system A') DEFINE CHANNEL(A.TO.B) + CHLTYPE(rcvr) REPLACE + TRPTYPE(tcp) + DESCR('Receiver channel from A to B') DEFINE QREMOTE('Queue2') REPLACE + RNAME('Queue2') + RQMNAME(SYSTEMA) + XMITQ(A) + DESCR('Queue 2 in system A') DEFINE QLOCAL(SYSTEMA) REPLACE + USAGE(xmitq) + DESCR('Xmit queue') DEFINE CHANNEL(B.TO.A) + CHLTYPE(sdr) REPLACE + TRPTYPE(tcp) CONNAME(WTR5317) + XMITQ(SYSTEMA) + DESCR('Sender channel from B to A')
File Name: SytemA.tst	File Name: SystemB.tst

- A receiver channel that receives messages and puts them into a local queue (B.TO.A in system A and A.TO.B in system B)
- A local queue from which the program gets its messages (Q2 in system A and Q1 in system B)

In each system, you must define the appropriate queue manager objects. The objects are defined in the two .tst files shown in Table 1.

For the AS/400, you may use MQ CL definition commands instead. Figure 11 on page 26 shows them for SystemB.

```
CRTMQMQ    QNAME('Queue1') QTYPE(*LCL) REPLACE(*YES) +
             TEXT('Messages from system A')

CRTMQMCHL  CHLNAME(A.TO.B) CHLTYPE(*RCVR) REPLACE(*YES) +
             TRPTYPE(*TCP) TEXT('Receiver channel from A to B')

CRTMQMQ    QNAME('Queue2') QTYPE(*RMT) REPLACE(*YES) +
             TEXT('Queue 2 in system A') +
             RMTQNAME('Queue2') RMTMQMNAME('SystemA') +
             TMQNAME(B.TO.A)

CRTMQMQ    QNAME(B.TO.A) QTYPE(*LCL) REPLACE(*YES) +
             TEXT('Xmit queue') USAGE(*TMQ)

CRTMQMCHL  CHLNAME(B.TO.A) CHLTYPE(*SDR) REPLACE(*YES) +
             TRPTYPE(*TCP) TEXT('Sender channel from B +
             to A') CONNAME(WTR5317) TMQNAME(B.TO.A)
```

Figure 11. AS/400 MQ CL Definitions

2.7.2 How to Start Communication Manually

First, the objects have to be known to the queue managers. Use RUNMQSC to create the objects. The OS/400 command is STRMQMMQSC. Make sure that the queue manager is running.

Next, start the listeners and the channels. You need to start only the sender channel in each system. MQSeries starts the receiver. The AS/400 listener is started with the command STRMQMLSR.

Table 2 on page 27 shows the commands to create objects for an application and to start communication. The scenario shows how to start the channels manually, and the application, too. In the next section, we show how to start channels automatically.

Notes:

1. How to start and stop AS/400 programs is discussed in detail in Chapter 6, “Running the Samples” on page 117.
2. In an AS/400 you can have only one queue manager.

Table 2. Start MQM-to-MQM Connection

Sender (System A)	Receiver (System B) AS/400
Do the following only once to create objects.	
`strmqm SYSTEMA` `runmqsc < systema.tst > a.a`	`STRMQM` `STRMQMMQSC SRCMBR(SYSTEMB) SRCFILE(MQLIB/QMQSC)`
Do this every time you establish connection.	

`strmqm SYSTEMA`
`start runmqlsr -t tcp`
`runmqsc`
`start channel(A.TO.B)`
`Ctrl+C`
	`STRMQM`
	`STRMQMLSR`
	
	`STRMQMCHL CHLNAME(A.TO.B)`

Now start the applications in both systems.	
Note: If you use the default queue manager you may omit the queue manager name (SYSTEMA) in the strmqm command.	

2.7.3 How to Start Channels Automatically

You can use the channel initiator to start channels. Instead of the commands shown in Table 2 enter the following commands (for Windows NT, UNIX and OS/2):

```
start runmqlsr -t tcp
start runmqchi
```

With the first command you start the listener and with the second the channel initiator program. The channel initiator monitors a channel initiation queue and starts the proper channel to read in the message. The default initiation queue is SYSTEM.CHANNEL.INITQ.

You may also start the channel initiator from RUNMQSC (Windows NT, UNIX and OS/2). The command is:

```
start chinit
--OR--
start chinit initq(SYSTEM.CHANNEL.INITQ)
```

In an AS/400 the channel initiator is started with this command:

```
STRMQMCHLI QNAME(SYSTEM.CHANNEL.INITQ)
```

Table 3 on page 28 shows the xmit queue definitions that start the channels automatically.

Table 3. MQSeries Definitions to Start Channels

Sender (System A)	Receiver (System B)
⋮ DEFINE QLOCAL(A.TO.B) REPLACE + USAGE(xmitq) + **TRIGGER TRIGTYPE(every) +** **INITQ(SYSTEM.CHANNEL.INITQ) +** DESCR('Xmit Queue')	⋮ DEFINE QLOCAL(B.TO.A) REPLACE + USAGE(xmitq) + **TRIGGER TRIGTYPE(every) +** **INITQ(SYSTEM.CHANNEL.INITQ) +** DESCR('Xmit queue')

For an AS/400, you may use this CL command instead:

```
CRTMQMQ QNAME(B.TO.A) QTYPE(*LCL)          +
        REPLACE(*YES) TEXT('Xmit queue')   +
        TRGENB(*YES) TRGTYPE(*ALL)         +
        INITQNAME(SYSTEM.CHANNEL.INITQ)
```

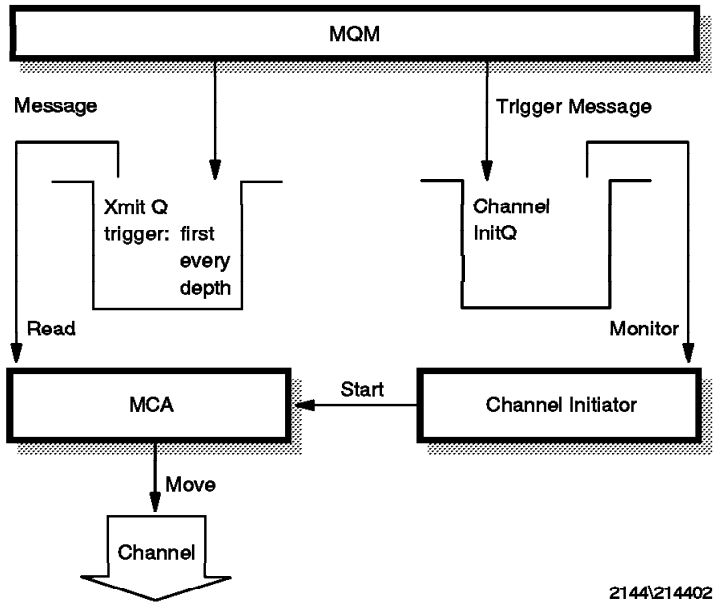


Figure 12. Triggering Channels

Figure 12 shows that the queue manager can trigger the process that starts the channel program in three ways:

- When the first message is put into the transmission queue (*FIRST)
- Every time a message is put into the xmit queue (*ALL)
- When the queue contains a specified number of messages (*DEPTH)

In order to trigger the channels you have to modify the .tst files shown in Table 1 on page 25. The changes are shown in bold in Table 3.

Since messages are placed into the xmit queue sporadically, we trigger the channel every time a message arrives. We use the default initiation queue for the trigger message.

2.7.4 How to Start Applications Automatically

This section describes how to trigger an application program that runs in the server machine. Both triggering and triggered applications can run in the same machine or in different machines.

Note: MQSeries for Windows V2.1 does not support triggering.

Figure 13 shows what happens when a message arrives from another system. The application program that has to process the message can be started manually from the command line or as part of the startup sequence. Another way is to let the queue manager start it when it is needed, that is, using the MQSeries triggering mechanism.

In this example, the listener program monitors the port assigned to MQSeries, 1414, and invokes the message channel agent (MCA) when a message arrives. The message is moved into the appropriate local queue (specified in the message header) and a trigger message is put into an initiation queue.

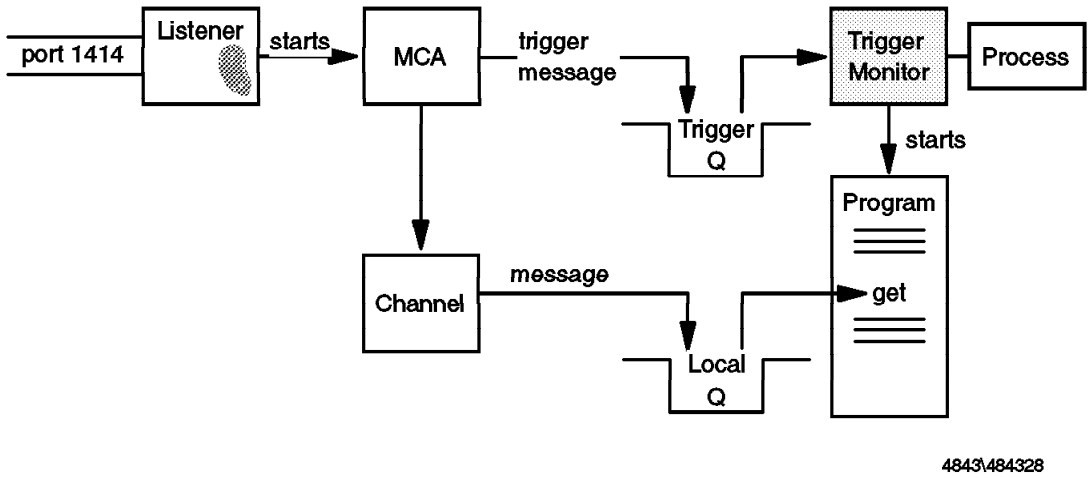


Figure 13. Triggering Applications

The trigger monitor monitors the initiation (trigger) queue and starts the program. The program name is specified in the process. The program then gets the message off the queue.

The next sections describe the MQSeries objects and API sequences in both sender and receiver systems.

2.7.5 How to Test if the Queue Manager Is Working

After you install MQSeries on the *server* you may want to check if the installation has been successful. You can do that with sample programs provided with MQSeries. In NT they are in the directory:

```
c:\mqm\tools\c\samples\bin
```

- AMQSPUT puts messages on a queue.
- AMQSGET gets messages from a queue.

AS/400 sample programs are available in source code in the library QMQMSAMP. Samples are written in the following languages:

Table 4. AS/400 Sample Programs and Programming Languages

Language	Source file	Program names
C	QCSRC	AMQSxxxx
RPG	QRPGSRC	AMQ1xxxx
ILE-RPG	QRPGLESRC	AMQ2xxxx
COBOL	QLBLSRC	AMQ0xxxx
ILE COBOL	QCBLLESRC	AMQ0xxxx

The programs have the following purpose:

- AMQxPUT4 puts three messages on a queue.
- AMQxGET4 gets all messages from a queue.
- AMQxGBR4 gets all messages from a queue without removing them.

So AMQ1PUT4 is the sample program in (non-ILE) RPG, that puts three messages on a queue.

Chapter 6, “Running the Samples” on page 117 deals with the sample programs in detail. The way the sample programs are written requires the library QMQM to be in the library list of the job executing the samples.

You may use one of the default queues when you run the sample programs. The example in Figure 14 on page 31 is for Windows NT. Type the commands shown in bold.

```
[C:\]strmqm
MQSeries queue manager started.
[C:\]cd mqm\tools\c\samples\bin
[C:\mqm\tools\c\samples\bin]amqsput SYSTEM.DEFAULT.LOCAL.QUEUE
Sample AMQSPUT0 start
target queue is SYSTEM.DEFAULT.LOCAL.QUEUE
111111
222222
333333
                    <--- 2 x Enter ends AMQSPUT
Sample AMQSPUT0 end

[C:\mqm\tools\c\samples\bin]amqsget SYSTEM.DEFAULT.LOCAL.QUEUE
Sample AMQSGET0 start
message <111111>
message <222222>
message <333333>
no more messages    <--- Wait for time out
Sample AMQSGET0 end

[C:\mqm\tools\c\samples\bin]endmqm /i MYQMGR
MQSeries queue manager ending.
MQSeries queue manager ended.

[C:\mqm\tools\c\samples\bin]
```

Figure 14. Testing the Queue Manager

Notes:

1. After AMQSPUT is started type a few messages and then press Enter twice to end it.
2. AMQSGET times out after a few seconds.

2.8 Communication between Client and Server

In this section, we discuss what you have to do to define and test the connection between an MQ client and its MQ server. A more detailed description is provided in the publication *MQSeries Clients*, GC33-1632.

Note: There is no Java client available for AS/400 Version 4 Release 2.

2.8.1 How to Define a Client/Server Connection

The following describes MQM objects and other definitions needed for a client/server connection. You have to know the transmission protocol (for

example, TCP/IP) and the addresses of the systems you want to connect, such as:

- 9.24.104.206 for the server
- 9.24.104.116 for the client

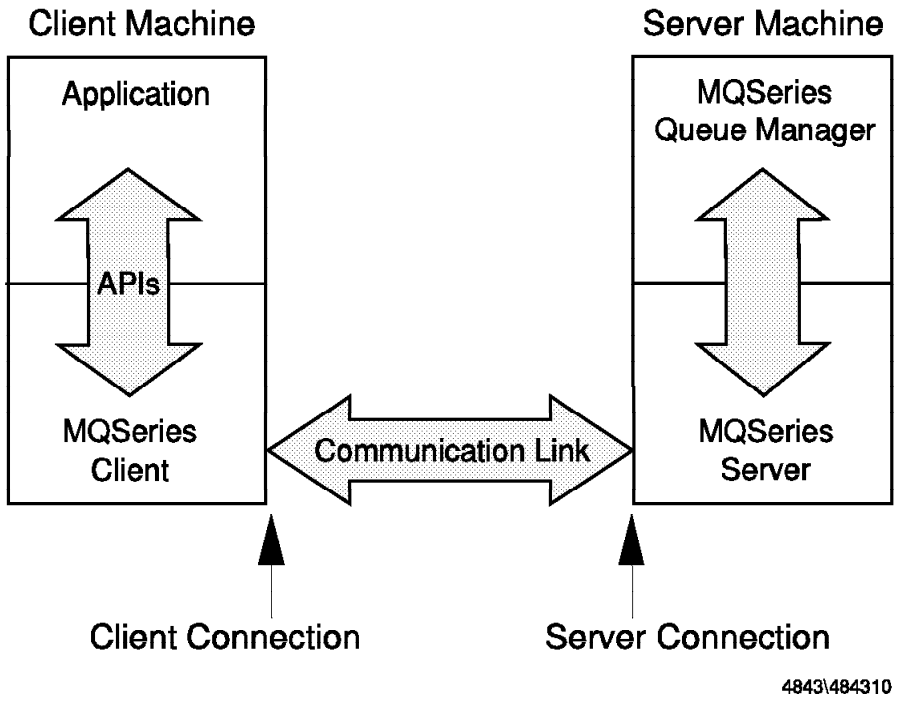


Figure 15. Client/Server Connection

2.8.1.1 Definition in the Server

Define the queues that the application needs and a channel of the type server connection. The queue manager definitions are in the file javacoma.tst, shown in Figure 16 on page 33.

We define a queue for the application to put messages in and an MQI channel of the type server connection. Create the objects by issuing the command:

```
runmqsc < javacoma.tst > a.a
```

```
*****************************************************************/
* File:  JAVACOMA.TST                                           */
*****************************************************************/

    DEFINE QLOCAL('JAVAQ1')  REPLACE +
           DESCR('Queue for Java Application')

    DEFINE CHANNEL('JAVACH1')  CHLTYPE(SVRCONN) REPLACE +
           TRPTYPE(TCP) MCAUSER(' ')
```

Figure 16. Definitions for Server Connection

2.8.1.2 Definitions in the Client

Define an environment variable for the MQSeries client that defines the connection on the client side. Set the variable with the following command or place the command in the CONFIG.SYS.

```
set MQSERVER=JAVACH1/TCP/9.24.104.206(1414)
```

Notes:

1. MQSERVER is the name of the environment variable.
2. JAVACH1 is the name of the channel to be used for communication between client and server. The channel must be defined in the server.
3. TCP denotes that TCP/IP is to be used to connect to the machine with the address following the parameter.
4. (1414) is the default port number for MQSeries. You may omit this parameter if the listener on the server side uses this default, too.

Java Client: For the MQSeries Client for Java, the environment variables are set in the applet code. An applet can run in any machine, such as a network station, and it has no access to a disk drive in the workstation. The example below shows what statements to include in your Java program:

```
import com.ibm.mq.*;
⋮
MQEnvironment.hostname = "9.24.104.456";
MQEnvironment.channel  = "JAVACH1";
MQEnvironment.port     = 1414;
⋮
```

2.8.2 How to Start a Client/Server Connection

Before you can start the application in the client you have to start a program in the server that listens to the communication link between client and server. MQSeries provides a program that does just that, the listener called runmqlsr.

You start the listener with the following command:

```
start runmqlsr /t tcp /m JAVAMQM /p 1414
```

Notes:

1. `start` creates a new window for the listener.
2. `runmqlsr` is the name of the listener program.
3. `/t tcp` defines that there is a TCP/IP connection between client and server.
4. `/m JAVAMQM` specifies the name of the queue manager the client connects to. If omitted, the default queue manager is used.
5. `/p 1414` defines the TCP/IP port number. 1414 is the default assigned to MQSeries applications; hence, you may omit this parameter.

The server is now ready to process MQI calls from the application running in the client.

2.8.3 How to Test a Client/Server Connection

With the steps described above communication between client and server is established. You can test the connection using programs provided with MQSeries. They are in the directory:

```
c:\mqm\tools\c\samples\bin
```

- AMQSPUTC puts messages on a queue.
- AMQSGETC gets messages from a queue.

Note: The ″C″ stands for client.

On the server machine, execute the following commands:

```
strmqm
start runmqlsr -t tcp
```

On the client machine, type the commands shown in bold in Figure 17 on page 35.

- After AMQSPUTC is started type a few messages and then press Enter twice to end it.
- AMQSGETC times out after a few seconds.

```
[C:\mqm\tools\c\samples\bin]amqsputc JAVAQ1
Sample AMQSPUT0 start
target queue is JAVAQ1
111111
222222
333333
                    <--- 2 x Enter ends AMQSPUTC
Sample AMQSPUT0 end

[C:\mqm\tools\c\samples\bin]amqsgetc JAVAQ1
Sample AMQSGET0 start
message <111111>
message <222222>
message <333333>
no more messages    <--- Wait for time out
Sample AMQSGET0 end

[C:\mqm\tools\c\samples\bin]
```

Figure 17. Testing Client/Server Connection

On the server machine, you see the following listener window after completion of the two programs:

```
RUNMQLSR.EXE

11/19/96  14:54:10 Channel program started
11/19/96  14:54:28 Channel program ended normally
11/19/96  14:55:20 Channel program started
11/19/96  14:55:49 Channel program ended normally
```

Figure 18. Listener Window (RUNMQLSR)

2.8.4 How a Client/Server Connection Works

This section describes how to trigger an application program that runs in the server machine. Since there are MQI channels of the type server connection between clients and server, all clients use the queue manager in the server machine. When a client puts a message on a queue it has to be read and processed by some program. This program can be started when the server starts or the queue manager can start it when needed by using the MQSeries triggering mechanism.

Figure 19 on page 36 shows two clients connected to a server. Both clients request services from the same program (Appl S1). Since that application runs in the same system as the queue manager, we have only local queues. Some queues are specifically for a particular client, for example, QA1 is the reply queue for client A and QA2 is the reply queue for client B. Other

queues are used by both clients and server. For example, QS1 is used as output queue for both clients and as input queue for the server program.

The next sections describe the MQSeries objects and API sequences in both client and server.

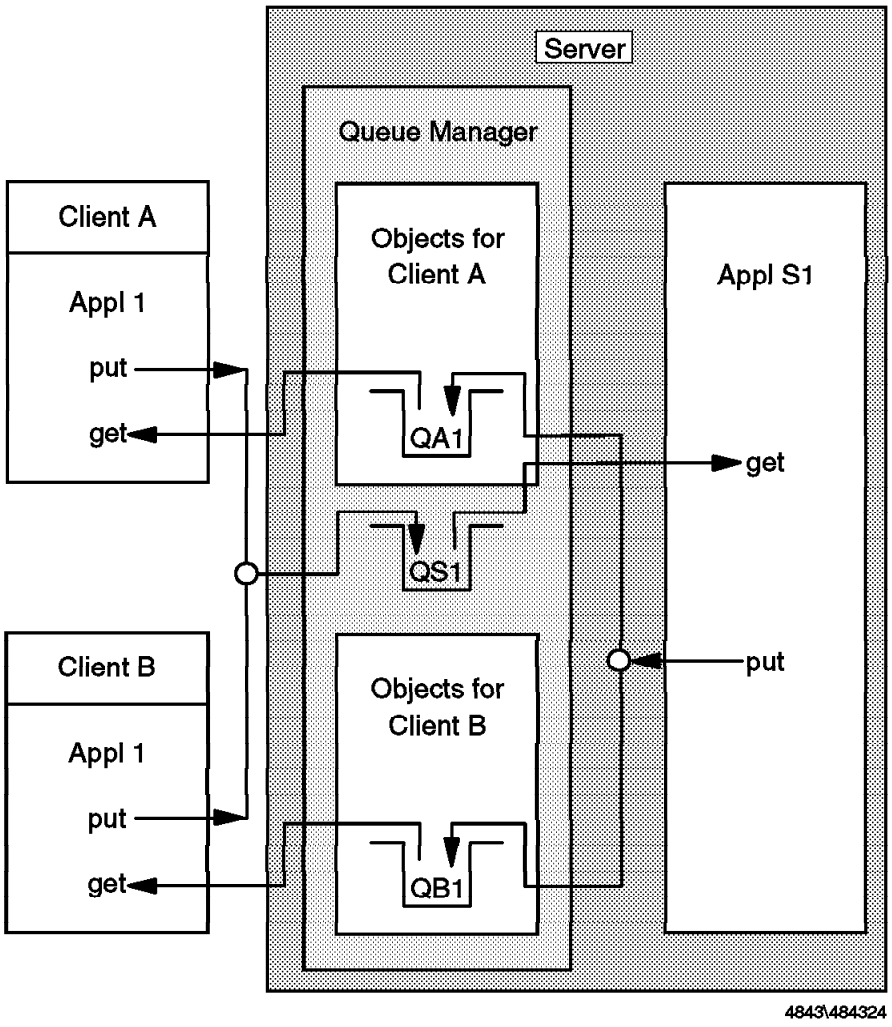


Figure 19. Clients and Server

2.8.4.1 How a Client Sends a Request

The client starts a program that puts a message on a queue. For this function five MQSeries API calls are executed:

- MQCONN connects to the queue manager in the server.
- MQOPEN opens the message queue for output (QS1).
- MQPUT puts the message on the queue.
- MQCLOSE closes the queue (QS1).
- MQDISC disconnects from the queue manager (JAVAMQM).

Of course, the program can put many messages in the queue before it closes it and disconnects. Closing the queue and disconnecting from the queue manager could be done when the application ends.

The MQSeries client code that runs in the client machine processes the API calls and routes them to the machine defined in the environment variable, such as:

```
set MQSERVER=JAVACH1/TCP/9.24.104.206
```

2.8.4.2 How the Server Receives a Request

In the server machine, the following queue manager objects are needed:

- A channel, JAVACH1, of the type server connection.
- A local queue, QS1, into which the clients put their messages.
- An initiation queue into which the queue manager puts a trigger message when a request for queue QS1 arrives. We use the default initiation queue.
- A process definition, process.S1, that contains the name of the program to be started when the trigger event occurs (S1).
- One or more queues in which the program puts the reply messages (QA1 and QB1).

The definitions for the objects are shown in Figure 20 on page 38.

In the server machine, two programs have to be started:

- The listener with the command `runmqlsr /t tcp`
- The trigger monitor with the command `start runmqtrm`

The listener listens for messages on the channel and puts them on the queue QS1. Since QS1 is triggered, the MQM puts a trigger message on the trigger queue each time a message is put on QS1. When a message is placed on the trigger queue, the trigger monitor starts the program defined in the process.

The server program S1 connects to the queue manager, opens the queue QS1 and issues an MQGET to read the message.

```
DEFINE CHANNEL('JAVACH1') CHLTYPE(SVRCONN) REPLACE +
       TRPTYPE(TCP) MCAUSER(' ')

DEFINE QLOCAL('QS1') REPLACE +
       DESCR('Queue for Server Application') +
       TRIGTYPE(EVERY) +
       TRIGGER INITQ (system.default.initiation.queue) +
       PROCESS (process.S1)

DEFINE PROCESS(process.S1) REPLACE +
       DESCR('Process to start server program') +
       APPLTYPE (OS2) +
       APPLICID('c:\mqtest\s1.exe')

DEFINE QLOCAL('QA1') REPLACE +
       DESCR('Reply queue for client A')

DEFINE QLOCAL('QB1') REPLACE +
       DESCR('Reply queue for client B')
```

Figure 20. Definitions for Two Clients and Server

2.8.4.3 How the Server Sends a Reply

After processing a request the server puts the reply in the reply queue for the client. To do this it has to open the output queue (QA1 or QB1) and issue an MQPUT.

Since several clients use the same server application, it is advisable to give the server a "return address," that is, the names of the queue and the queue manager that will receive the reply message. These fields are in the header of the request message, containing the reply-to-queue manager and reply-to-queue (here, QA1 or QB1). It is the responsibility of the client program to specify these values.

Usually, the server program stays active and waits for more messages, at least for a certain time. For how long can be specified in the wait option of the MQGET API.

2.8.4.4 How the Client Receives a Reply

The client program knows the name of its input queue, here QA1 or QB1. The application can use two modes of communication:

- Conversational

  If the application uses this mode of communication with the server program, it waits for the message to arrive before it continues processing. This means, the reply queue is open and an MQGET with wait option has been issued.

The client application must be able to deal with two possibilities:

- The message arrives in time.
- The timer expires and no message is there.

- True asynchronous

  When using this mode, the client does not care when the request message arrives. Usually, the user clicks on a push button in a menu window to activate a program that checks the reply queue for messages. If a message is present, this or another program can process the reply.

2.9 The Message Queuing Interface (MQI)

A program talks directly to its local queue manager. It resides in the same processor or domain (for clients) as the program itself. The program uses the Message Queuing Interface (MQI). The MQI is a set of API calls that request services from the queue manager.

Note: When the connection between a client and its server is broken, no APIs can be executed, since all objects reside in the server.

There are 13 APIs. The most important ones are:

MQPUT Put a message on a queue.

MQGET Get a message from a queue.

The other calls are used less frequently:

MQCONN Establishes connection with a queue manager using the standard bindings.

MQCONNX Establishes connection with a queue manager using fastpath bindings. Fastpath puts and gets are faster, but the application must be well behaved. Application and queue manager run in the same process. When the application crashes it takes the queue manager down with it. This API call is new in MQSeries Version 5.

MQBEGIN This API begins a unit of work that is coordinated by the queue manager, and that may involve external XA compliant resource managers. This API has been introduced with MQSeries

Version 5. It is used to coordinate transactions that use queues (MQPUT and MQGET under syncpoint) and database updates (SQL commands).

MQOPEN Open or obtain access to a queue.

MQCLOSE Close a queue.

MQDISC Disconnect from the queue manager.

MQPUT1 Open a queue, put a message on it and close the queue. This is a combination of MQOPEN, MQPUT and MQCLOSE.

MQINQ Request information about the queue manager or one of its objects, such as the number of messages in a queue.

MQSET Change some attributes of a queue.

MQCMIT A syncpoint has been reached. Messages put as part of a unit of work are made available to other applications. Messages retrieved as part of a unit of work are deleted.

MQBACK The queue manager has to back out all message puts and gets that have occurred since the last syncpoint. Messages put as part of a unit of work are deleted. Messages retrieved as part of a unit of work are reinstated on the queue.

Note: MQDISC implies the commit of a unit of work. Ending the program without disconnecting from the queue manager causes a rollback (MQBACK).

MQSeries for AS/400 does not use MQBEGIN, MQCMIT or MQBACK. The commit control operation codes of the AS/400 language are used.

In CL language these commands are available:

Table 5. AS/400 Commit/Rollback Commands

Start commit control	STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
Commit:	COMMIT
Rollback:	ROLLBACK
End commit control:	ENDCMTCTL

Before executing ENDCMTCTL, the application must commit or rollback, and then disconnect from the MQ Manager. The command to disconnect is DSCMQM.

2.9.1 A Code Fragment

The code fragment in Figure 21 on page 42 shows the APIs to put a message on one queue and get the reply from another queue.

Note: The fields CompCode and Reason will contain completion codes for the APIs. You can find them in the *Application Programming Reference*.

1. This statement connects the application to the queue manager with the name MYQMGR. If the parameter QMName does not contain a name, then the default queue manager is used. MQ stores in the variable HCon the handle of the queue manager. This handle must be used in all subsequent APIs.
2. To open a queue the queue name must be moved into the object descriptor that will be used for that queue. This statement opens QUEUE1 for output only (open option MQOO_OUTPUT). Returned are the handle to the queue and values in the object descriptor. The handle Hobj1 must be specified in the MQPUT.
3. MQPUT places the message assembled in a buffer on a queue. Parameters for MQPUT are:
   - The handle of the queue manager (from MQCONN)
   - The handle of the queue (from MQOPEN)
   - The message descriptor
   - A structure containing options for the put (refer to the *Application Programming Reference*)
   - The message length
   - The buffer containing the data
4. This statement closes the output queue. Since the queue is predefined no close processing takes place (MQOC_NONE).
5. This statement opens QUEUE2 for input only using the queue-defined defaults. You could also open a queue for browsing, meaning that the message will not be removed.
6. For the get, the nowait option is used. The MQGET needs the length of the buffer as input parameter. Since there is no message ID or correlation ID specified, the first message from the queue is read.

   You may specify a wait interval (in milliseconds) here. You can check the return code to find out if the time has expired and no message arrived.
7. This statement closes the input queue.
8. The application disconnects from the queue manager.

```
MQHCONN  HCon;                  // Connection handle
MQHOBJ   HObj1;                 // Object handle for queue 1
MQHOBJ   HObj2;                 // Object handle for queue 2
MQLONG   CompCode, Reason;      // Return codes
MQLONG   options;
MQOD     od1 = {MQOD_DEFAULT};  // Object descriptor for queue 1
MQOD     od2 = {MQOD_DEFAULT};  // Object descriptor for queue 2
MQMD     md  = {MQMD_DEFAULT};  // Message descriptor
MQPMO    pmo = {MQPMO_DEFAULT}; // Put message options
MQGMO    gmo = {MQPMO_DEFAULT}; // Get message options
:
// 1 Connect application to a queue manager.
strcpy  (QMName,"MYQMGR");
MQCONN  (QMName, &HCon, &CompCode, &Reason);

// 2 Open a queue for output
strcpy  (od1.ObjectName,"QUEUE1");
MQOPEN  (HCon,&od1, MQOO_OUTPUT, &Hobj1, &CompCode, &Reason);

// 3 Put a message on the queue
MQPUT   (HCon, Hobj1, &md, &pmo, 100, &buffer, &CompCode, &Reason);

// 4 Close the output queue
MQCLOSE (HCon, &Hobj1, MQCO_NONE, &CompCode, &Reason);

// 5 Open input queue
options = MQOO_INPUT_AS_Q_DEF;
strcpy  (od2.ObjectName, "QUEUE2");
MQOPEN  (HCon, &od2, options, &Hobj2, &CompCode, &Reason);

// 6 Get message
gmo.Options = MQGMO_NO_WAIT;
buflen  = sizeof(buffer - 1);
memcpy (md.MsgId, MQMI_NONE, sizeof(md.MsgId);
memset (md.CorrelId, 0x00, sizeof(MQBYTE24));
MQGET (HCon, Hobj2, &md, &gmo, buflen, buffer, 100, &CompCode, &Reason);

// 7 Close the input queue
options = 0;
MQCLOSE (HCon, &Hobj2,options, &CompCode, &Reason);

// 8 Disconnect from queue manager
MQDISC  (HCon, &CompCode, &Reason);
```

Figure 21. Fragments of an MQSeries Program

Communication between programs is *time-independent*. The sender can continue processing without waiting for a reply. The receiving program does not even have to run. MQSeries holds the messages until the applications are ready to process them.

MQSeries applications are *message-driven*. The arrival of a message triggers an event. Just like clicking on a push button in a GUI invokes some procedure, a message can trigger the program that processes the message data.

Chapter 3. AS/400 for Beginners

If you have a PC background and are looking for the ″dir″ and ″path″ in an AS/400, the following is for you. If you are familiar with the AS/400 and its operating system, skip this chapter.

In the following sections, we explain how to sign on, start programs, use AS/400 commands, and how to edit, compile and test your own programs.

3.1 Signing On

Figure 22 shows the AS/400 sign on panel. Usually, you type only your user ID and password.

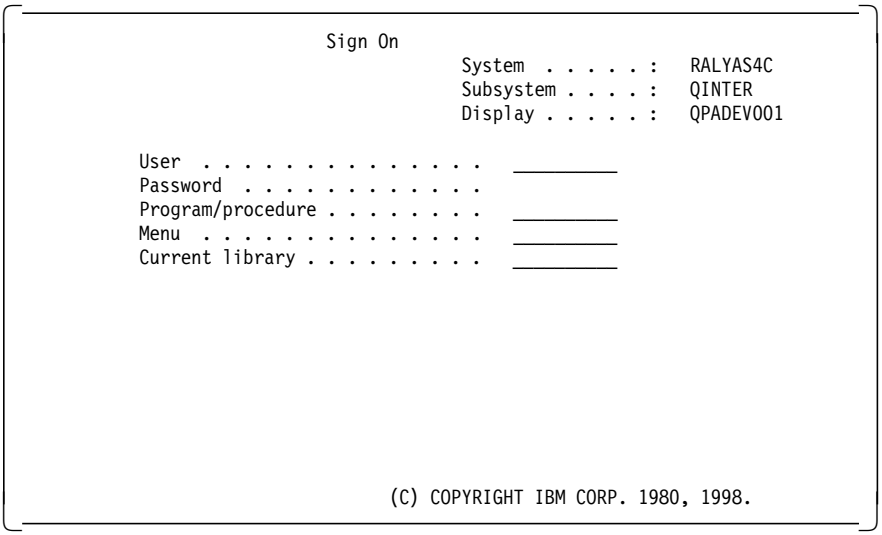


Figure 22. AS/400 Sign On Panel

The user (ID) is really the name of a profile, an AS/400 object of the type USRPRF (user profile). The user profile contains information about the type of jobs a user is allowed to execute, such as security officer, programmer, operator and user. Each AS/400 object, such as program or file, has a list of users that can work with it.

No user job can start without a user profile.

Note: Data entered is not case sensitive.

© Copyright IBM Corp. 1998

3.2 Main Menu and Command Entry Panel

The main menu shown in Figure 23 can guide you to many AS/400 functions. In this book, however, we ask you to type commands on the command line instead of using the menu.

```
MAIN                              AS/400 Main Menu
                                                          System:   RALYAS4C
Select one of the following:

     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. Client Access/400 tasks

    90. Sign off

Selection or command
===> dsplibl
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
```

Figure 23. AS/400 Main Menu

If you enter the command **dsplibl** as shown in Figure 23, another panel appears that displays the library list. What that is we will discuss later. To return the the main menu, press either F3 or F12.

If you prefer to see the *history of commands* you entered instead of returning to the main menu after each command, type **call qcmd** at the command line and press Enter. This displays the Command Entry panel shown in Figure 24 on page 47.

From both panels, Main Menu and Command Entry panel, you can retrieve the last entered commands with the function key F9. When you use the Command Entry panel, you can also scroll using the Page Up and Page Down keys and position the cursor at a command before pressing F9.

You can return to the main menu with F12.

You can also create a new copy of the main menu by entering **go main**. Don't work with too many invocation levels (call, call, go, call, etc.); you may find it difficult to remember with which of the main menus you were working with.

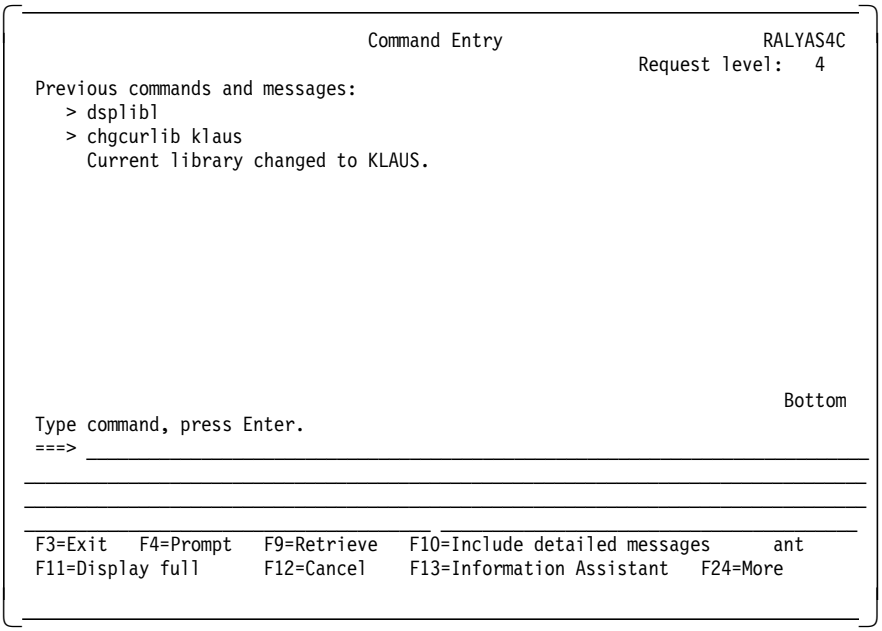


Figure 24. AS/400 Command Entry Panel

3.3 Understanding the 5250 Screen Layout

You may be familiar with the 3270, but the 5250 interface of the AS/400 is different, even the keyboard is different. PA keys don't exist and what you think is a "New Line" is a "Field Exit". It erases data in a field from the cursor position to the end of the field and moves the cursor to the next field on the screen. Pressing Clear does not make sense, because the AS/400 will just redisplay your latest screen panel.

The panel in Figure 25 on page 48 is an example of a 5250 layout.

1. Input fields are underlined.
2. Function keys are always referred to with the prefix "F", such as F3=Exit.

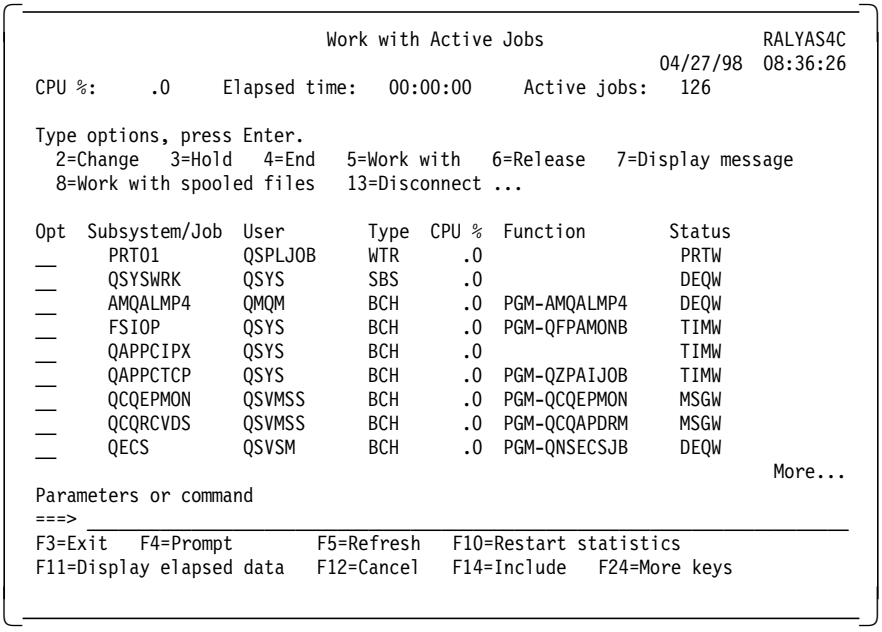


Figure 25. Work with Active Jobs Panel

3. Numbers alone are not function keys, they are *options*. You type your option in the underscored field in front of the choice you select, for example, 2=Change and 3=Hold.
4. If you see the text More.. on the right side of the panel, you can use the Page Up and Page Down keys to view information on the previous or following page, respectively. Don't use F7 and F8.
5. F1 is the help key. The help text depends on the cursor position.
6. The sign = = = > at the bottom indicates the beginning of the command line.

   Note: If the command line is only two positions long, your user profile is not allowed to enter commands. You will have to get the authority from a security officer.

The AS/400 also has a Windows 95/NT look alike interface, but this does not yet apply to MQSeries administration and a majority of other AS/400 applications.

3.4 Using Commands

In this section, we explain how to use AS/400 commands. As an example, we create a job description. When the operating system starts a job, it looks here for attributes, such as what libraries to use. Instead of creating your own, you may also use a default job description.

In this book, we may tell you to enter a command like this:

```
CRTJOBD JOBD(ABCJOB) INLLIBL(ABCLIB QGPL QTEMP)
```

However, you don't have to key the command exactly like above, you can get away with less typing. It is just a short way for us to instruct you what to do. Instead, you may follow these steps:

Step 1. Type only the command and the first parameter in lowercase:

crtjobd abcjob

Step 2. Press the prompt key F4.

This brings up the Create Job Description panel shown in Figure 26.

```
                      Create Job Description (CRTJOBD)

 Type choices, press Enter.

 Job description  . . . . . > ABCJOB____    Name
   Library  . . . . . .        *CURLIB___   Name, *CURLIB
 Job queue  . . . . . .       QBATCH____    Name
   Library  . . . . . .        *LIBL_____   Name, *LIBL, *CURLIB
 Job priority (on JOBQ) . . . .  5          1-9
 Output priority (on OUTQ)  . . .  5         1-9
 Print device . . . . . .       *USRPRF___   Name, *USRPRF, *SYSVAL...
 Output queue . . . . . .       *USRPRF___   Name, *USRPRF, *DEV, *WRKSTN
   Library  . . . . . .         __________   Name, *LIBL, *CURLIB
 Text 'description' . . . . . .  *BLANK_____________________________________
_____




                                                                      Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display       F24=More keys
```

Figure 26. Create Job Description Panel

Step 3. Now press F11 to get a different view of the panel. Figure 27 on page 50 shows that the keywords for the parameters are now

displayed, too. Here you can fill in the data for the fields that are displayed.

Note: You may not see F11 on the bottom of your panel.

In our example, you see that the keyword for the job description JOBD is followed by the parameter ABCJOB, just as you typed it in the command `crtjobd abcjob`.

```
                      Create Job Description (CRTJOBD)

Type choices, press Enter.

Job description  . . . . . JOBD         > ABCJOB____
  Library  . . . . . .                     *CURLIB___
Job queue  . . . . . JOBQ                 QBATCH____
  Library  . . . . . .                     *LIBL_____
Job priority (on JOBQ) . . . . . JOBPTY   5
Output priority (on OUTQ)  . . . OUTPTY   5
Print device . . . . . PRTDEV             *USRPRF___
Output queue . . . . . OUTQ               *USRPRF___
  Library  . . . . . .                     __________
Text 'description' . . . . . . TEXT       *BLANK_____________________________
_____________________


                                                                        Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure 27. Create Job Description Panel - Keywords

```
                      Create Job Description (CRTJOBD)

 Type choices, press Enter.

 Print text . . . . . . . . . . . PRTTXT         *SYSVAL_____________________________
_
 Accounting code  . . . . . . . . ACGCDE         *USRPRF_____________
 Routing data . . . . . . . . . . RTGDTA         QCMDI_______________________________
_____________________________________________
 Request data or command  . . . . RQSDTA         *NONE_______________________________
_____________________________________________________________________________________
_____________________________________________________________________________________
_____________________________________________________________
 CL syntax check  . . . . . . . . SYNTAX         *NOCHK
 Initial library list . . . . . . INLLIBL        *SYSVAL___
                          + for more values      +_________
 End severity . . . . . . . . . . ENDSEV         30____



                                                                          More...
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 28. Create Job Description Panel - More Keywords

Step 4. You also have to type three values in the initial library list (INLLIBL). Remember the command:

CRTJOBD JOBD(ABCJOB) INLLIBL(ABCLIB QGPL QTEMP)

Where do you find the keyword INLLIBL?

Step 5. If some keywords do not show up and the panel cannot be scrolled, then press F10 for more keywords and scroll forward using Page Down. The word More**...** on the bottom right side of the panel indicates that there are more pages to view. Figure 28 shows the second page of the panel after F10 has been pressed.

Step 6. You can toggle with F11 between the view with and without the keywords shown.

Step 7. Just fill in the parameters for the keywords as instructed.

Some keywords may be dependent on data entered, and will only appear after you fill in some fields.

- If you blank out a parameter and press Enter the default for that field will be displayed again.
- If you overwrite data with '?' followed by a blank you will get a new panel showing the options for that keyword.

- Pressing F1 displays the help text for the keyword where the cursor is positioned. If the cursor is at the top of the panel the help text is for the command.

Step 8. In our example, we have to enter three values for the keyword INLLIBL. However, the panel shows only space for two. What do we do now?

Step 9. Enter the + sign in the field following the text + for more values and press Enter.

Step 10. This brings up the panel shown in Figure 29. Here you enter the names of the three libraries as shown in bold.

Step 11. Press Enter to return to the previous panel.

```
                  Specify More Values for Parameter INLLIBL

Type choices, press Enter.

Initial library list . . . . . .   abclib____
                                   qgpl_____
                                   qtemp____
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                   __________
                                                                      More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 29. Panel to Specify More Parameters

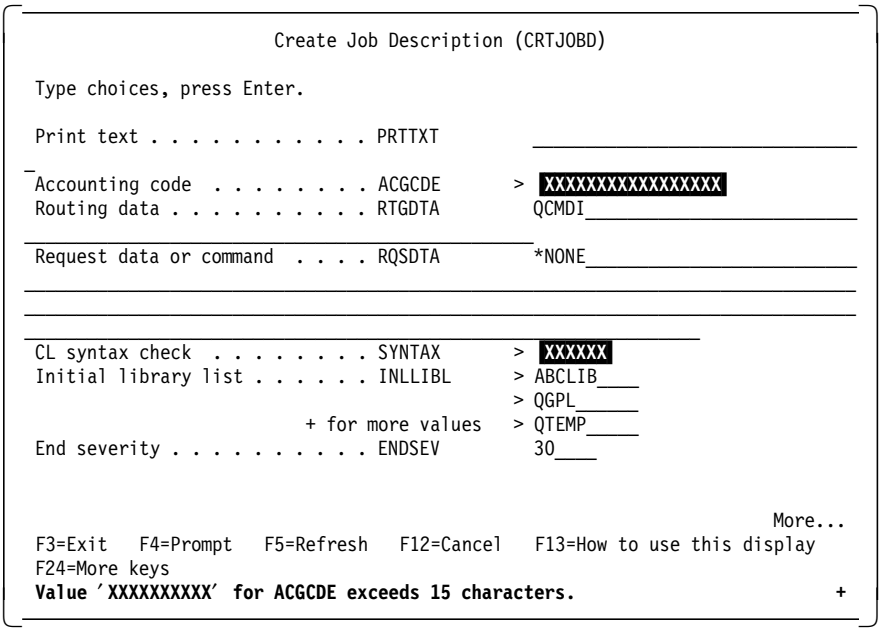


Figure 30. Panel with Errors

3.4.1 Input Errors

Fields that contain an error are displayed in reverse image. Figure 30 shows an example. Error messages are displayed at the bottom of the panel. If there is more than one error message you will see a + sign at the bottom right. To scroll the through messages, position the cursor at the message and press Page Down. If you want more message text, then position the cursor in the message and press F1.

3.4.2 Executing a Command

Press Enter to execute the command. After the command has been executed successfully, the command line will be blank. Otherwise, the command in error will remain on the command line and an error message is displayed on the bottom of the panel.

3.4.3 What Do I Do When Something Goes Wrong?

If you made an error and created an object with wrong data, simply delete the object with the appropriate delete command and then create a new one. Here are examples of create and delete commands:

- crtjobd / dltjobd
- crtrpgpgm / dltpgm

- crtmqmq / dltmqmq

In our example, to delete the job description we just created type:

```
dltjobd abcjob
```

If you made an error changing an object with a change command, change it back using the same change command.

3.4.4 Shortcuts

If you don't know the complete name of a command then you may enter the first characters followed by an asterisk. The system will then display all the commands that begin with the letters you typed.

For example, if you type `crtmqm*` and press Enter you will see the Select Command panel shown in Figure 31. It displays all the MQSeries for AS/400 commands that create objects.

To select a command, type 1 in front of one of the commands and press Enter. The subsequent panel prompts you for the parameters that must be entered.

```
                                Select Command

Type options, press Enter.
  1=Select

Opt  Command     Library     Text
 _   CRTMQM      QSYS        Create Message Queue Manager
 _   CRTMQMCHL   QSYS        Create MQM Channel
 _   CRTMQMPRC   QSYS        Create MQM Process
 1   CRTMQMQ     QSYS        Create MQM Queue









                                                                      Bottom
Parameters or command
===> ____________________________________________________________________________
F3=Exit      F4=Prompt   F5=Refresh   F9=Retrieve   F11=Display names only
F12=Cancel   F16=Repeat position to   F17=Position to
```

Figure 31. Select Command Panel

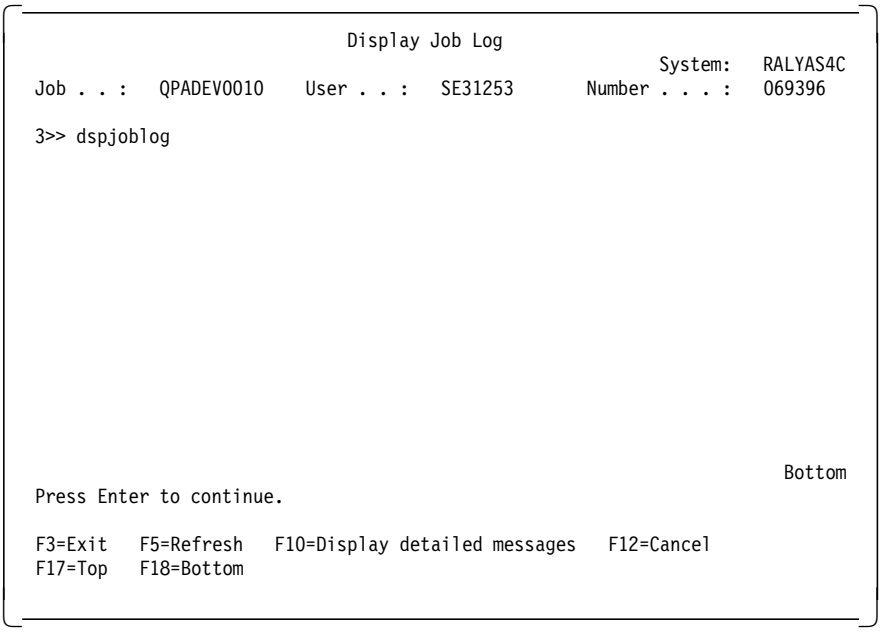


Figure 32. Display Job Log Panel

3.5 Where Is the Error Log?

As long as a job is active the command history and system messages sent to the job can be displayed with the *display job log* command. If you are signed on to the job follow these steps:

Step 1. Type the command dspjob.

Step 2. Select 10 (Display job log, if active or on job queue). This displays the panel shown in Figure 32.

Step 3. Then you must press F10 to display the Display All Messages panel shown in Figure 33 on page 56. True, there is not a big difference between the two panels, but now you can *scroll back* and read the job log.

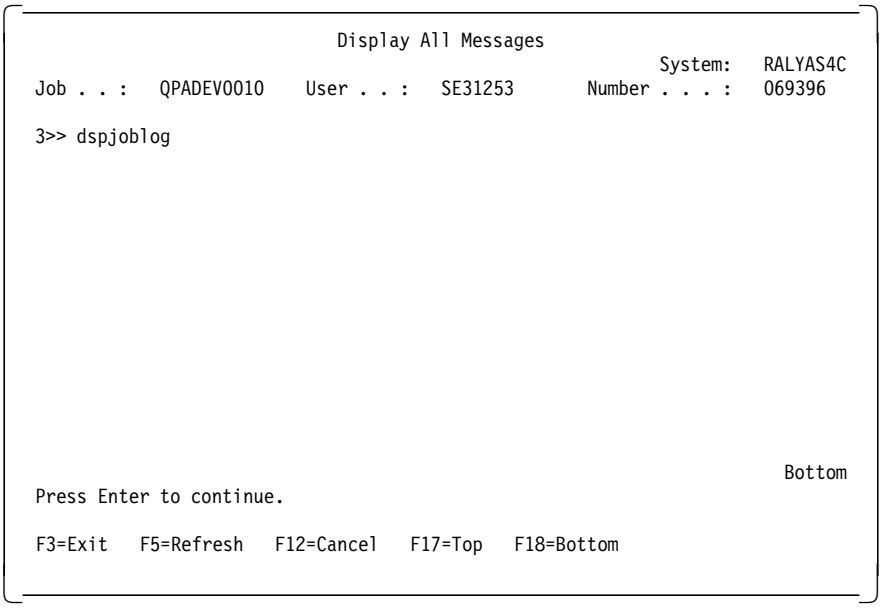


Figure 33. Display All Messages Panel

When a job has finished executing the job log is written to the "outqueue" QEZJOBLOG. You can display the log with the command:

```
WRKOUTQ OUTQ(QEZJOBLOG)
```

You get an overwiew of all available job logs and you must select the one you want to display. If the job has the log attribute *NOLIST, writing the job log to QEZJOBLOG at end of job will be suppressed.

3.6 About Libraries

Is there a DIR command?

The OS/400 does not know a dir command; however, there is a similar command named *Work with Object Links*, WRKLNK. When you enter this command, the Work with Object Links panel is displayed. It shows the AS/400 Internal File System (IFS). You will see that it is very much like UNIX and PCs. Page down and you will see a panel similar to that shown in Figure 34 on page 57.

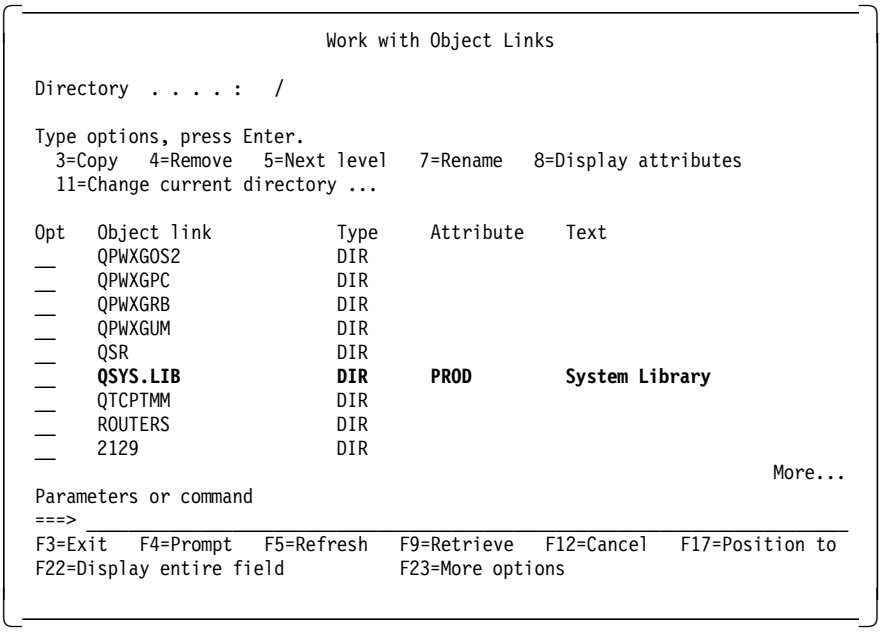


Figure 34. Work with Object Links Panel

In the examples in this book, however, we will not use this file system. The IFS is used when the AS/400 is used for specific functions, such as a server for PCs or network stations or as an HTTP server.

The database, executable programs, and all traditional AS/400 objects reside in what you see in Figure 34 as QSYS.LIB.

The traditional (non-IFS) commands refer to the ″System Library″ as QSYS.

The System Library contains system objects and all other libraries, such as the MQSeries libraries. All AS/400 objects must be in a library.

Only the System Library can hold other libraries. Libraries can be user libraries holding application programs and files, or they can be IBM libraries containing compilers or other licensed program products.

So in the normal (non-IFS) environment you have only two levels:

- The System Library named QSYS with the OS/400 operating system.
- All other libraries. IBM delivered libraries start with the letter Q, such as QGPL, QRPG, QMQM, etc.

There are more than 20 different object types. Some of them are programs, commands, files, data areas, user space and configuration objects.

If you want to browse the AS/400 objects in the traditional way display the libraries using:

```
DSPLIB
```

3.6.1 Using Program Development Management (PDM)

If you have the Program Development Management software (5769-PW1) installed you can browse the libraries using the following PDM commands:

WRKLIBPDM	Work with Libraries Using PDM
WRKOBJPDM	Work with Objects Using PDM
WRKMBRPDM	Work with Members Using PDM

Let us look at some command sequences and find out how PDM works. The examples use MQSeries libraries; however, you can choose other libraries, QGPL for instance.

Step 1. We will execute all commands from the Command Entry panel (Figure 24 on page 47). If you are in the main menu type the following command to get to it:

```
call qcmd
```

Step 2. To look at all objects in the system, try the following command:

```
WRKOBJPDM LIB(QSYS) OBJ(*ALL) OBJTYPE(*ALL)
```

It will display a panel that lists all objects of all types in the system library. If you scroll through the panel you see objects of over 40 types, such as LIB, USRPRF, CTLD, PGM, FILE and JOBD. This list contains too many objects. Let us display the objects of one type only.

Step 3. To get back to the Command Entry panel press Enter.

```
                          Work with Objects Using PDM                 RALYAS4C

Library . . . . .   QSYS______       Position to . . . . . .   __________
                                     Position to type  . . . . .   __________

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display       7=Rename
  8=Display description        9=Save       10=Restore      11=Move ...

Opt  Object      Type        Attribute   Text
__   QMQM        *LIB        PROD        MQSERIES FOR AS/400
__   QMQMADM     *LIB        PROD        MQSERIES FOR AS/400 - ADMIN APPLICATI
__   QMQMDATA    *LIB        PROD        MQSERIES FOR AS/400 - OBJECTS
__   QMQMPROC    *LIB        PROD        MQSERIES FOR AS/400 - PROCESSES
12   QMQMSAMP    *LIB        PROD



                                                                       Bottom
Parameters or command
===> ____________________________________________________________________________
F3=Exit          F4=Prompt           F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry   F23=More options    F24=More keys
This is a subsetted list.                                                    +
```

Figure 35. Work with Objects Using PDM Panel

```
                          Work with Objects Using PDM                 RALYAS4C

Library . . . . .   QMQMSAMP__       Position to . . . . . .   __________
                                     Position to type  . . . . .   __________

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display       7=Rename
  8=Display description        9=Save       10=Restore      11=Move ...

Opt  Object      Type        Attribute   Text
__   AMQIEX54    *PGM        CLP
__   AMQ1GBR4    *PGM        RPG         Sample using Browse
__   AMQSDATA    *FILE       PF-DTA      Input Data Samples
__   QCBLLESRC   *FILE       PF-SRC      ILE COBOL/400 Samples
__   QCLSRC      *FILE       PF-SRC      CL Samples
__   QCSRC       *FILE       PF-SRC      C Samples
__   QLBLSRC     *FILE       PF-SRC      COBOL Samples
__   QMQSC       *FILE       PF-SRC      MQSC Samples
                                                                       More...
Parameters or command
===> ____________________________________________________________________________
F3=Exit          F4=Prompt           F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry   F23=More options    F24=More keys
```

Figure 36. Work with Objects Using PDM Panel

Step 4. To look at *all libraries* in the system, enter this command:

```
WRKOBJPDM LIB(QSYS) OBJ(*ALL) OBJTYPE(*LIB)
```

Note: You can press F9 to retrieve the previously entered command and modify it.

Step 5. To see all libraries in the system that begin with QMQ, enter this command:

```
WRKOBJPDM LIB(QSYS) OBJ(QMQ*) OBJTYPE(*LIB)
```

The Work with Objects Using PDM panel shown in Figure 35 on page 59 shows all libraries starting with qmq.

Step 6. Now we want to see what one of the libraries contains. In AS/400 terms, we want to *work with* members in the library. This is option 12.

The panel shows only options up to 11. For more options press F23, that is Shift+11.

Type 12 (work with) in front of QMQMSAMP and all objects in the library will be displayed. Figure 36 on page 59 shows the first part of the panel. You can scroll the list.

Note: You can even run a command or a program from this panel. To do this, select option 16 and then press F4.

```
                          Work with Members Using PDM                RALYAS4C

File  . . . . . .   QRPGLESRC_
  Library . . . .     QMQMSAMP__           Position to  . . . . .   _________

Type options, press Enter.
 2=Edit         3=Copy  4=Delete 5=Display       6=Print     7=Rename
 8=Display description  9=Save  13=Change text  14=Compile  15=Create module...

Opt  Member      Type        Text
__   AMQ2ECH4    RPGLE_____  Sample echo program_____________________________
__   AMQ2GBR4    RPGLE_____  Sample using Browse_____________________________
5_   AMQ2GET4    RPGLE_____  Sample using MQGET_______________________________
__   AMQ2INQ4    RPGLE_____  Sample using MQINQ_______________________________
__   AMQ2PUT4    RPGLE_____  Sample using MQPUT_______________________________
__   AMQ2REQ4    RPGLE_____  Sample using reply queue_____________________________
__   AMQ2SET4    RPGLE_____  Sample using MQSET_____________________________
__   AMQ3ECH4    RPGLE_____  SAMPLE ECHO PROGRAM_____________________________
                                                                       More...
Parameters or command
===> ____________________________________________________________________
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
```

Figure 37. Work with Members Using PDM Panel

Step 7. Now scroll forward and type 12 (work with) in the options field in front of the file QRPGLESRC, a file that contains the ILE RPG/400 samples, and press Enter. This brings up the panel shown in Figure 37.

Step 8. Now type 5 (option display) in front of the source member AMQ2GET4 and press Enter. This displays the Browse panel shown in Figure 38. You can scroll to browse this RPG source file. To return to the previous panel press Enter or F12 (cancel).

```
 Columns . . . :    1  71          Browse                  QMQMSAMP/QRPGLESRC
 SEU==> ____________________________________________________________  AMQ2GET4
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       *************** Beginning of data ************************************
0001.00      H
0002.00       ****************************************************************
0003.00       *                                                              *
0004.00       * Program name: AMQ2GET4                                       *
0005.00       *                                                              *
0006.00       * Description: Sample RPG program that gets messages from      *
0007.00       *              a message queue (example using MQGET)           *
0008.00       *                                                              *
0009.00       * Statement:   Licensed Materials - Property of IBM            *
0010.00       *                                                              *
0011.00       *              5763-MQ2                                        *
0012.00       *              (C) Copyright IBM Corporation 1994, 1996        *
0013.00       *                                                              *
0014.00       * Status: Version 3 Release 2.0                                *
0015.00       *                                                              *
0016.00       ****************************************************************

 F3=Exit   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle   F12=Cancel
 F16=Repeat find        F24=More keys
                                     (C) COPYRIGHT IBM CORP. 1981, 1998.
```

Figure 38. Browse Panel

Note: Option 2 invokes the Source Edit Utility (SEU) which allows you to edit the file. This function should be left to your AS/400 programmer.

Step 9. You can also browse a ″data file″. Data files cannot be changed as easily as ″source files″.

Return to the Work with Objects Using PDM panel shown in Figure 36 on page 59 by pressing F12.

Note: Do not use F3; that takes you back to the very beginning.

Step 10. At the panel for the library QMQMSAMP, type 12 (the work with option) in front of the file AMQSDATA. The data file has the attribute PF_DATA while the source file has the attribute PF-SRC.

```
                        Display Physical File Member
File . . . . . :   AMQSDATA           Library  . . . . :   QMQMSAMP
Member . . . . . :   INQ              Record . . . . . :   1
Control  . . . . .                    Column . . . . . :   1
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
SYSTEM.SAMPLE.INQ
SYSTEM.SAMPLE.LOCAL
SYSTEM.SAMPLE.SET
SYSTEM.SAMPLE.INQ
SYSTEM.SAMPLE.NONEXIST

                       ****** END OF DATA ******









                                                                        Bottom
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

Figure 39. Display Physical File Member Panel (1)

```
                        Display Physical File Member
File . . . . . :   AMQSDATA           Library  . . . . :   QMQMSAMP
Member . . . . . :   INQ              Record . . . . . :   1
Control  . . . . .                    Column . . . . . :   1
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
SYSTEM.SAMPLE.INQ
EEEECD4ECDDDC4CDD4444444444444444444444444444444444444444444444444444444444444444
282354B214735B958000000000000000000000000000000000000000000000000000000000000000

SYSTEM.SAMPLE.LOCAL
EEEECD4ECDDDC4DDCCD4444444444444444444444444444444444444444444444444444444444444444
282354B214735B363130000000000000000000000000000000000000000000000000000000000000000

SYSTEM.SAMPLE.SET
EEEECD4ECDDDC4ECE4444444444444444444444444444444444444444444444444444444444444444
282354B214735B253000000000000000000000000000000000000000000000000000000000000000000

SYSTEM.SAMPLE.INQ
EEEECD4ECDDDC4CDD4444444444444444444444444444444444444444444444444444444444444444
282354B214735B958000000000000000000000000000000000000000000000000000000000000000000
                                                                        More...
F10=Display character   F11=Display side-by-side   F24=More keys
```

Figure 40. Display Physical File Member Panel (2)

The next panel you see shows the following four members in the ″partitioned dataset″:

```
___  ECHO   ECHO sample input data
__5  INQ    INQ sample input data
___  PUT    PUT sample input data
___  SET    SET sample input data
```

Note: There is no edit option in the panel.

Type 5 (display option) in front of the member INQ to browse the content which is shown in Figure 39 on page 62.

Step 11. Now explore the use of function keys F24 (Shift + F12), F10 and F11. Pressing F10 and then F11 displays the panel shown in Figure 40 on page 62.

3.7 About Library Lists

Is it a ″path″?

No, not exactly, it is rather a search list.

You can refer to an object by pointing to a specific library, for example:

```
DSPOBJD OBJ(QGPL/QRPGSRC) OBJTYPE(*FILE)
```

Or you can refer to an object without specifying a library:

```
DSPOBJD OBJ(QRPGSRC) OBJTYPE(*FILE)
```

Without a library specified, the system will search for the object in the sequence of the libraries in the Library List, referred to as **LIBL* in the commands.

The library list for your job is viewed with the command DSPLIBL.

The Library List consists of four parts:

- SYS - System Library List
- PRD - Product Library
- CUR - Current Library
- USR - User Library List

3.7.1 System Library List

The System part is controlled by a system value which you may display with the command:

```
DSPSYSVAL SYSVAL(QSYSLIBL)
```

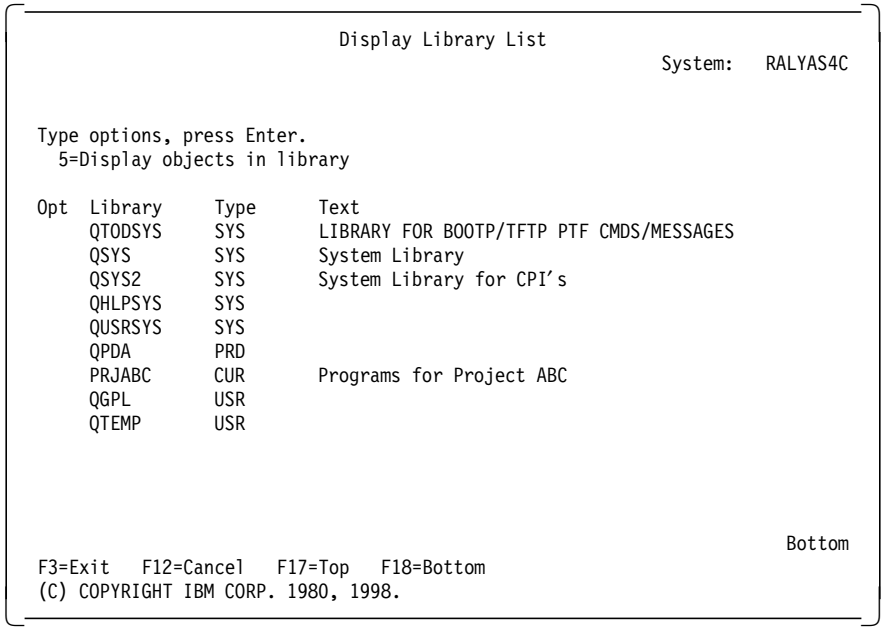


Figure 41. Display Library List

Only security officers should be allowed to change system values.

Note: You should not put an application library in the system library list.

MQSeries for AS/400 has a *quiesce* program that ends all jobs with a *lock* on QMQM. If QMQM is present in the system value QSYSLIBL, all jobs in the system will have QMQM in its library list, and the quiesce program will terminate all jobs in the system.

If you use a secondary language, insert the secondary language library in the system library list in front of QSYS. This command will allow Swedish to be used as secondary language:

```
CHGSYSLIBL LIB(QSYS2937)
```

Today, the most common situation outside the USA, is that you have your national language objects in the library QSYS. To see the US English text for commands and system messages, you insert the library QSYS2924.

You revoke the secondary language (here Swedish) with the command:

```
CHGSYSLIBL LIB(QSYS2937)
```

3.7.2 Product Library

You will not always see a product library displayed at DSPLIBL.

The product library is controlled by the operating system alone. The product library is inserted when OS/400 needs to refer to libraries that are holding licensed code that is not stored in the QSYS library.

If you start program development with the command STRPDM, and display the library list (DSPLIBL), you will see that the product library QPDA is inserted.

3.7.3 Current Library

Without a current library the create commands of OS/400 create objects in the library QGPL (General Purpose Library). This is probably not what you want.

To change your current library to PRJABC, for example, enter the command:

```
CHGCURLIB CURLIB(PRJABC)
```

To make the change permanent every time you use your user profile ABC, this command can be executed by the security officer:

```
CHGUSRPRF USRPRF(ABC) CURLIB(PRJABC)
```

3.7.4 User Library List

The user library list is controlled by the system value QUSRLIBL. Usually QGPL and QTEMP are included.

QGPL is a general all purpose library, so you should put here objects of general interest to all users.

QTEMP is a special type of object. A unique QTEMP library exists for every job in the system. This library and its content disappear after the job has terminated.

The user library list should contain the application libraries for the user.

This is done by executing the following command:

```
CHGLIBL LIBL(QGPL QTEMP APPLIB1) CURLIB(XXLIB)
```

This command also controls the current library.

To display the Edit Library List panel shown in Figure 42 on page 66 enter the following command:

```
edtlibl
```

The best way to control the user library list and other job attributes is to describe them in a ″Job Description″ and refer to this job description in the user profile:

```
CRTJOBD JOBD(ABC/ABCUSER) INLLIBL(ABCLIB QGPL QTEMP)
CHGUSRPRF USRPRF(ABCUSER) JOBD(ABC/ABCUSER)
```

This an example. You should not change user profiles or create job descriptions without assistance from someone responsible for AS/400 security and backup/recovery.

```
                          Edit Library List
                                                                RALYAS4C
Type new/changed information, press Enter.
  To add a library, type name and desired sequence number.
  To remove a library, space over library name.
  To change position of a library, type new sequence number.


Sequence                  Sequence                  Sequence
 Number    Library        Number    Library         Number    Library
  010      __________      120      __________       230      __________
  020      QGPL_____       130      __________       240      __________
  030      QTEMP____       140      __________       250      __________
  040      __________      150      __________
  050      __________      160      __________
  060      __________      170      __________
  070      __________      180      __________
  080      __________      190      __________
  090      __________      200      __________
  100      __________      210      __________
  110      __________      220      __________
                                                              Bottom
F3=Exit           F5=Refresh            F12=Cancel
```

Figure 42. Edit Library List

3.8 More Job Attributes and Where They Come From

Job attributes can be displayed with the display job command shown below:

```
DSPJOB
```

An overview of all jobs can be displayed with the display active jobs command as follows:

```
DSPACTJOB
```

3.8.1 General Job Attributes

General job attributes are specified in the *system values*. System values are displayed interactively with the command:

```
WRKSYSVAL
```

Note: Only a security officer can change system values.

3.8.2 Specific Job Attributes

Specific job attributes are described in *user profiles* and in *job descriptions*.

Note: Only a security officers can create and change user profiles.

3.8.2.1 Create User Profile Example

The following is an example of how to create a user profile:

```
CRTUSRPRF USRPRF(ABCUSER) PWDEXP(*YES)  USRCLS(*PGMR) CURLIB(ABCLIB)
          INLPGM(ABCLIB/ABCPGM) INLMNU(*SIGNOFF) LMTCPB(*YES)
          TEXT('just an example') JOBD(ABCLIB/ABCJOB) GRPPRF(ABCGRP)
          OWNER(*GRPPRF) SUPGRPPRF(PAYROLL)
```

If certain values are not specified in the user profile, system values will be used, for example, values for country and coded character set.

If the user profile does not refer to a specific job description it will use the *default job description*.

Examples of job attributes specified in a job description:

- Initial library list
- Output queue
- Message logging
- Log CL program commands
- Allow multiple threads

3.8.2.2 Create Job Description Example

A job description can be created with the following command:

```
CRTJOBD JOBD(ABCLIB/ABDJOB) OUTQ(ABCLIB/ABCOQ) TEXT('just a sample')
        INLLIBL(ABCLIB QGPL QTEMP) LOG(4 0 *SECLVL) LOGCLPGM(*YES)
        ALWMLTTHD(*YES)
```

The default job description is in the library QGPL.

You can view the default job description with the command:

```
DSPJOBD JOBD(QDFTJOBD)
```

3.8.3 When Is It Decided What Job Atttributes to Use

The job attributes are determined at job start, but can be changed during execution with the command:

```
CHGJOB
```

When a batch job is started from an interactive job (submitted), most job attributes (library list, user profile, etc.) are inherited from the submitting job. So called routing entries in the (batch) subsystem generally change the run behaviour (priority). The submit command (SBMJOB) may also change job attributes.

3.9 Where Is the Editor?

When you write a program you create a file member in a source file. A source file is created with the command *create source physical file*:

```
CRTSRCPF FILE(ABCSRC/QRPGLESRC) TEXT('ILE RPG Source code ABC')
```

The AS/400 Source Edit Utility adds members to the source file.

You may edit source code with any editor, such as Notepad at your PC. But then you must transfer the code to the source file by ClientAccess file transfer or cut-and-paste, before the compilation.

The most common method to edit source code in AS/400, is to use the 5250 interface with the *Source Edit Utility (SEU)*, that comes with the Licensed Program Product *5769-PW1 Application Development ToolSet for AS/400.* The Application Development ToolSet for AS/400 is often referred to as PDM, which stands for Program Development Manager.

SEU is similar to mainframe 3270 applications such as XEDIT and EPM. SEU can do syntax checking and prompting while entering the source code, provided you have told the editor which type code you are entering.

Your AS/400 programmer will probably invoke the editor with the command:

```
WRKMBRPDM
```

This will prompt him to the source file he previously used.

The example in Figure 43 shows source code that obviously has been copied from the samples in QMQMSAMP/QRPGSRC. It is always wise not to change the originals.

The programmer may also invoke this panel from the main menu via the following options:

5 Programming
2 Programming Development Manager (PDM)
3 Work with members

```
                          Work with Members Using PDM                RALYAS4C

File  . . . . . .   QRPGSRC___
  Library . . . .     MYMQLIB___          Position to  . . . . .   __________

Type options, press Enter.
 2=Edit         3=Copy  4=Delete 5=Display       6=Print     7=Rename
 8=Display description  9=Save  13=Change text  14=Compile  15=Create module...

Opt  Member      Type        Text
__   AMQ1ECH4    RPG_______  Sample echo program_____________________________
__   AMQ1GBR4    RPG_______  Sample using Browse_____________________________
__   AMQ1GET4    RPG_______  Sample using MQGET_____________________________
__   AMQ1INQ4    RPG_______  Sample using MQINQ_____________________________
__   AMQ1PUT4    RPG_______  Sample using MQPUT_____________________________
__   AMQ1REQ4    RPG_______  Sample using reply queue_____________________
__   AMQ1SET4    RPG_______  Sample using MQSET_____________________________

                                                                       Bottom
Parameters or command
===> ____________________________________________________________________________
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

Figure 43. Work with Members Using PDM

There are some conventions on how to name the source files. If they are followed you will have an easier time when you edit and compile. Here are some examples:

QCBLLESRC for ILE COBOL/400

QCLSRC for CL

QCSRC for C

QLBLSRC	for COBOL
QMQSC	for MQSC
QRPGLESRC	for ILE RPG/400
QRPGSRC	for RPG
QDDSSRC	for Files (physical, logical, display, and print files)
QTBLSRC	for Tables
QCMDSRC	for Commands

If you are editing C language code and your keyboard is not set up for US English, you should pay attention to the CCSID for the source file. You probably want to key the characters { } as they appear on your keyboard and not as your precious national letters. And you would also want the compiler to understand that you have entered them as C language characters.

CCSID for a file can be viewed by entering the command:

```
DSPFD FILE(QMQMSAMP/QCSRC)
```

You can change CCSID by this command:

```
CHGSRCPF FILE(MYLIB/MYCSRC) CCSID(278)
```

The redbook *Speak the Right Language with Your AS/400 System*, SG24-2154, can help you get this right.

Remember that you may run into problems, if your source code is a copy of a member from another source file with a different CCSID.

It is not our intention to solve these problems here. Your programmer has to resolve this.

By entering 2 in front of the source code you start the editor, and F6 helps you to create a new (empty) member.

While editing F4 prompting is often used to do syntax checking while entering the code. Syntax check is provided for the common AS/400 languages.

Once again, it is not our intention to teach you program development, but to give you an idea of what is happening.

There is also a richer client/server interface to program development, *5763CL2, ADTS Client Server for AS/400*. However, describing application

development in detail is not in the scope of this book. You can learn about this in IBM courses and you can be assisted by IBM consultants, IBM business partners, or certified AS/400 programmers.

```
 Columns . . . :    1  71           Edit                       QMQMSAMP/QRPGSRC
 SEU==> _____________________________________________________________  AMQ1GET4
 FMT *  ..... *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0130.00       *                                                               *
0131.00       *   Get messages from message queue                             *
0132.00       *                                                               *
0133.00       ***************************************************************
0134.00       **  initial loop condition based on result of MQOPEN
0135.00      C                    MOVE OCODE     CCODE
0136.00       *   buffer length available ...
0137.00      C                    Z-ADD60        BUFLEN
0138.00       **  start of loop to read all the messages . . .
0139.00      C           CCODE    DOWNECCFAIL
0140.00       * option is to wait up to 15 seconds for next message
0141.00      C                    Z-ADDGMWT      GMOPT           wait
0142.00      C                    ADD  GMCONV    GMOPT           convert
0143.00      C                    Z-ADD15000     GMWI            up to 15sec
0144.00       *
0145.00       **  MsgId and CorrelId are selectors that must be cleared
0146.00       **  to get messages in sequence, and they are set each MQGET

 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
```

Figure 44. Edit an RPG Program

3.10 How Do I Compile?

Your programmer will probably enter 14 in the line for the source code at the PDM menu and then press F4 in order to prompt the create command.

The command will then be submitted to batch.

Some source programs refer to other sources (with /copy statements) or to other objects (like display or print files). If this is the case, your library list should contain the library that contains the referred object.

The examples provided with MQSeries for AS/400 refer to source files in the library QMQM. The code describes the MQ message data structures. This code should either be copied to your application library, or you should include the library QMQM in the user library list during program development.

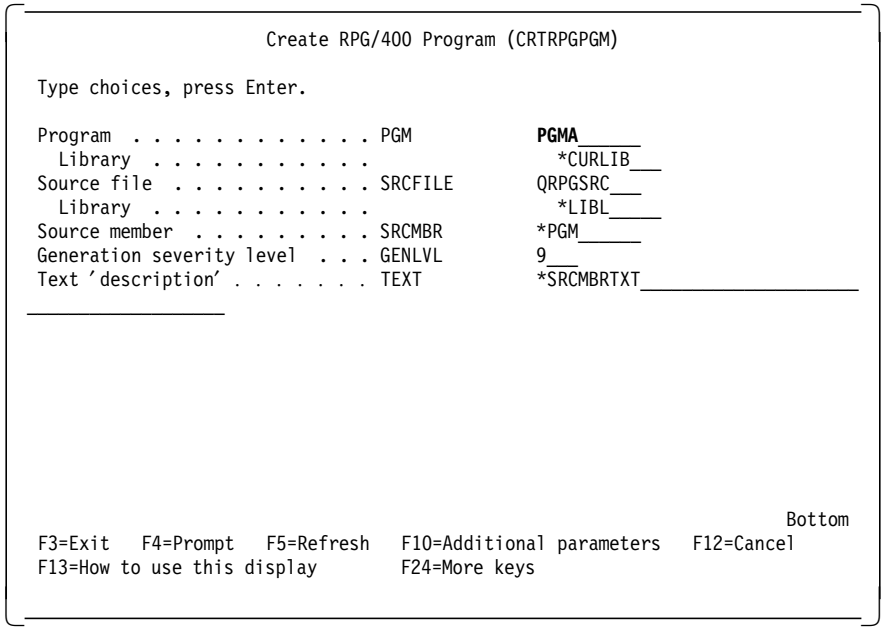


Figure 45. Create a Program Panel

Traditional RPG programs are compiled by the command below. If you use the F4 prompt you get the panel shown in Figure 45 on page 72.

CRTRPGPGM PGM(PGMA)

As you can see, the program object is compiled into the *current library* (*CURLIB). The name of the file member that contains the program will be the name of the program object compiled (PGMA). The text entered for file member is also moved to the program object.

The following command will also send the compilation to batch:

```
SBMJOB CMD(CRTRPGPGM PGM(PGMA))
```

When you program in the *ILE environment* (ILE RPG, ILE Cobol, ILE C) you will probably compile the source to *modules* and them combine (*bind*) the modules into programs or *service programs*.

Service programs will require a source member for specifying EXPORT and IMPORT. The importance of an ILE Environment is performance. Combining many modules into one program object causes less program calls and helps you to control file open/close. ILE also makes it possible to combine

different languages in the same program object, and after testing and debugging you can optimize your modules and programs without recompiling them.

You may ask your programmer if he/she is familiar with these functions.

Java

Java is evolving very quickly in the AS/400 world. However, at the writing of this book there are no MQSeries Java classes available for the AS/400.

3.11 Did the Program Compile OK?

Entering WS on the option line in the Work with Members Using PDM panel displays the jobs that you have submitted to batch. You can see whether they are in the queue, executing or finished.

Typing DM shows the messages produced by the compiler so that you can see whether or not the compile was successful.

When you are in editing mode with SEU, F15 can help you browsing the compiler listing.

The commands WRKSPLF or WRKOUTQ can also help you locate the printout.

3.12 Where Is the Printout?

Often a printer located somewhere in your area has already printed the compiler listing. In order to be more effective and to save all that paper, we recommend that you create your own *output queue* and let the printout stay there and display it on a screen.

Create your own output queue with the following command:

```
CRTOUTQ OUTQ(MYLIB/TEMPOUT)
```

Make it the output queue for printouts for your job and the jobs that you submit:

```
CHGJOB OUTQ(MYLIB/TEMPOUT)
```

You can browse the output with this command:

```
WRKOUTQ OUTQ(MYLIB/TEMPOUT)
```

Then type 5 (display option) next to the name of the listing as shown in Figure 46 on page 74.

Note: The command Work with Spooled Files, WRKSPLF, also allows you to browse the printout files.

```
                          Work with Output Queue

Queue:   OUT            Library:   DIETER         Status:   RLS

Type options, press Enter.
  1=Send   2=Change   3=Hold   4=Delete   5=Display   6=Release   7=Messages
  8=Attributes        9=Work with printing status

Opt  File        User        User Data   Sts   Pages   Copies  Form Type   Pty
 _   QSYSPRT     SE31253                 RDY       1        1   *STD         5
 _   AMQ1PUT4    DIETER                  RDY      12        1   *STD         5
 _   AMQ1PUT4    DIETER                  RDY      46        1   *STD         5
 _   QSYSPRT     DIETER      AMQ1PUT4    RDY       2        1   *STD         5
 _   QSYSPRT     DIETER      AMQ1PUT4    RDY       2        1   *STD         5
 5   AMQ1GBR4    DIETER                  RDY      46        1   *STD         5
 _   AMQ1GBR4    DIETER                  RDY      46        1   *STD         5
 _   QSYSPRT     DIETER      AMQ1GBR4    RDY       2        1   *STD         5

                                                                        Bottom
Parameters for options 1, 2, 3 or command
===> ____________________________________________________________________________
F3=Exit   F11=View 2   F12=Cancel   F20=Writers   F22=Printers
F24=More keys
```

Figure 46. Work with Output Queue Panel

3.13 Basic AS/400 Security

AS/400 security is based on the fact that everything a user works with is an *object*, and the operating system will always check user access to an object before a job can work with it.

An object can be a physical file, a program, a logical file, a spool file, a device description, or one of more than 20 other types.

No job can start in AS/400 without a user ID (user profile in AS/400 terms). Online jobs are initiated by signon. Batch jobs are submitted by a user. Automatically started jobs are initiated by an operator or a programmer. Jobs initiated from other systems must present the name of a user profile known to the AS/400 that is supposed to do the work.

All objects have a set of attributes. The most common are: name, library where the object is stored, and authority.

The authority in an object tells the operating system which users are allowed to work with the object and what they can do with it.

All users can be referred to as *public.

The user that creates an object automatically has **all* object authority to this object as he is the **owner*.

The object may also point to an *authority list*. Adding users to and deleting users from the authority list makes user access administration simpler.

In short: The permission to access an object is an attribute of the object. It is not an attribute of the user.

```
                          Edit Object Authority

Object . . . . . :   SRC             Owner  . . . . . :   KLAUS
  Library  . . . . :     KLAUS       Primary group  . . :   *NONE
Object type  . . . :   *FILE

Type changes to current authorities, press Enter.

  Object secured by authorization list  . . . . . .   *NONE_____

                        Object    ----------Object-----------
User        Group       Authority  Opr  Mgt  Exist  Alter  Ref
KLAUS                   *ALL____    X    X     X      X     X
*PUBLIC                 *CHANGE_    X    _     _      _     _









                                                                  Bottom
F3=Exit   F5=Refresh   F6=Add new users     F10=Grant with reference object
F11=Display data authorities   F12=Cancel   F17=Top   F18=Bottom
```

Figure 47. Edit Object Authority - Example 1

Figure 47 and Figure 48 on page 76 show two examples of the *grant object authority* command that changes the authority for an object. In order to execute this command you need authority to operate (Opr) the command GRTOBJAUT, and the authority to manage (Mgt) the object.

Security changes are normally done by the *Object Owner*, and often by somebody signed on with a security officer's profile (QSECOFR)

```
                          Edit Object Authority

Object . . . . . :   SRC            Owner  . . . . . :   KLAUS
  Library  . . . . :     KLAUS       Primary group  . . :   *NONE
Object type  . . . :   *FILE

Type changes to current authorities, press Enter.

  Object secured by authorization list  . . . . . .  *NONE_____

                         Object    ---------------Data---------------
User        Group        Authority  Read  Add  Update  Delete  Execute
KLAUS                    *ALL____    X     X     X       X        X
*PUBLIC                  *CHANGE_    X     X     X       X        X

                                                                   Bottom
F3=Exit   F5=Refresh    F6=Add new users   F10=Grant with reference object
F11=Nondisplay detail   F12=Cancel          F17=Top   F18=Bottom
```

Figure 48. Edit Object Authority - Example 2

3.14 Basic AS/400 Backup and Recovery

There are numerous redbooks and conventional AS/400 manuals about this subject. We recommend a new redbook, *MQSeries, Backup and Recovery*, SG24-5222, available in 3Q98.

In this section, we just summarize a few points, and we do not at all discuss disaster planning, dual systems, RAID and mirror technique, nor recovery of licensed internal code.

In order to operate backup/recovery (save/restore) you need to be signed on as a system operator (QSYSOPR). Specific functions need the authority of a security officer (QSECOFR).

Backup and recovery often require certain operating conditions; for example, objects should not be in use by other users when saved or restored.

3.14.1 Securing Data with Backups

In order to secure data you need to take offline copies (backups) to a safe place on a safe media (tape).

OS/400 has menus for general save/restore. Here are some backup or save commands:

SAVSYS Saves the internal licensed code and the OS/400 operating system

SAVLICPGM Saves IBM-supplied licensed programs

*SAVLIB LIB(*NONSYS)* Saves all libraries not holding IBM licensed code.

SAVLIB Saves a library (and its contents)

SAVOBJ Saves the objects only in a library

SAVCHGOBJ Saves only objects that have changed since last save

SAVSECDTA Saves security data (also in SAVSYS)

SAVCFG Saves communication objects (also in SAVSYS)

SAVSTG Saves all disk data

SAVDLO Saves OfficeVision documents

SAV Saves objects stored in IFS

SAVRSTxxx Set of commands that save from one AS/400 and restore on another

SAVOBJ can execute in a ″save-while-active″ mode. This increases processing time for backup and for the application that is touched.

Most save commands have a corresponding restore command, such as SAVLIB and RSTLIB, SAVOBJ and RSTOBJ. SAVSYS is an exception.

3.14.2 Securing Data between Backups

Many AS/400 installations do not save data between major backups. Needless to say that this is not sufficient. Data loss is not prevented by disk-RAID or disk-mirroring only. Files may be corrupted by the operator or application misbehavior, not to mention system errors.

Journal management is the standard way for AS/400 to save data between backups. If used effectively you should not need a daily backup of data. You may even rely on a rolling backup schedule, taking only a part of the data offline every day.

Journal management is based on two AS/400 object types:

Journals Journals have information about what files are saved and where the record image of changed records is stored.

Journal receivers

Receivers contain the images of the changed records (before and after image).

The following example shows:

- How to set up a journal
- How to use a file to simulate a disaster
- How to recover when a file is corrupted
- How to recover when a journal is damaged

3.14.2.1 Setting Up a Journal

With the following commands you create a journal and assign it to a file:

```
CRTLIB LIB(MYAPP) TEXT('test application')
CRTLIB LIB(MYJRN) TEXT('jrn, jrnrcv for MYAPP')
CRTLIB LIB(MYSAV) TEXT('save files for MYAPP')
CRTSRCPF FILE(MYAPP/SRC) TEXT('source file')
CRTJRNRCV JRNRCV(MYJRN/RCV)
CRTJRN JRN(MYJRN/JRN) JRNRCV(MYJRN/RCV)
CRTSAVF FILE(MYSAV/SVAPP)
CRTSAVF FILE(MYSAV/SVJRN)
```

Let your programmer key the source for the file FILA and compile it to MYAPP. Then set the file under journal management:

```
CRTPF FILE(MYAPP/FILA) SRCFILE(MYAPP/SRC)
STRJRNPF FILE(MYAPP/FILA) JRN(MYJRN/JRN) IMAGES(*BOTH)
```

A sample of FILA is in A.2, "A Simple File: FILA" on page 187.

3.14.2.2 Creating a File to Simulate a Disaster

The following shows how to create a file and put some records into it:

- Use the PDM with the command STRPDM.
- Select 1 (Work with libraries).
- In the subsequent panel type MYAPP as library name.
- Type 18 (Change using DFU) in front of FILA.
- Enter some records of your choice.

You can display the content of the file with the command:

```
DSPPFM FILE(MQAPP/FILA)
```

You can display the content of the journal receiver with the command:

```
DSPJRN JRN(MYJRN/JRN)
```

With the file under journal management you can simulate certain distaster and recovery scenarios. Use savefiles instead of a tape.

APYJRNCHG Apply entries from the receivers to your files. Use this after you have restored a backup.

RMVJRNCHG Takes away from the file the updates, deletes, and puts of records within the range of receiver sequences that you indicate in the command.

3.14.2.3 Disaster Scenario: Corrupt File

You will have to reply ″ignore″ (I know what I am doing) to some messages when you simulate a disaster by deleting objects. Here are the commands that simulate a recovery due to a deleted file:

```
SAVLIB LIB(MYAPP) DEV(*SAVF) SAVF(MYSAV/SVAPP)
SAVLIB LIB(MYJRN) DEV(*SAVF) SAVF(MYSAV/SVJRN)
STRPDM              /* and add some more records to your file FILA */
DSPPFM FILE(MYAPP/FILA)                    /* look at the records  */

DLTF FILE(MYAPP/FILA)                                  /* disaster */

RSTOBJ OBJ(FILA) SAVLIB(MYAPP) DEV(*SAVF) SAVF(MYSAV/SVAPP)
DSPPFM FILE(MYAPP/FILA)           /* watch the level from the save */
APYJRNCHG JRN(MYJRN/JRN) FILE((MYAPP/FILA))
DSPPFM FILE(MYAPP/FILA)     /* all the latest records are in place */
DSPJRN JRN(MYJRN/JRN)       /* watch the JRN entries for all this  */
```

Don't let the message from APYJRNCHG mislead you. Read it and you will see that it just tells you when APYJRNCHG came to a stop.

3.14.2.4 Disaster Scenario: Damaged Application and Journal

You will have to reply ″ignore″ (I know what I am doing) to some messages when you simulate a disaster by deleting objects. The commands that simulate a recovery due to a deleted journal and application follow:

```
SAVLIB LIB(MYAPP) DEV(*SAVF) SAVF(MYSAV/SVAPP)
SAVLIB LIB(MYJRN) DEV(*SAVF) SAVF(MYSAV/SVJRN)
STRPDM                              /* add some more records to FILA */
DSPPFM FILE(MYAPP/FILA)                       /* look at the records  */

DLTLIB LIB(MYAPP)                                          /* disaster */
DLTLIB LIB(MYJRN)                                          /* disaster */

RSTLIB SAVLIB(MYJRN) DEV(*SAVF) SAVF(MYSAV/SVJRN)
                     /* Journal object MUST be restored before files */
RSTLIB SAVLIB(MYAPP) DEV(*SAVF) SAVF(MYSAV/SVJRN)
                           /* restore application, journaling starts */
DSPPFM FILE(MYAPP/FILA)              /* watch the level from the save */
APYJRNCHG JRN(MYJRN/JRN) FILE((MYAPP/FILA))
DSPPFM FILE(MYAPP/FILA)        /* all the latest records are in place */
DSPJRN JRN(MYJRN/JRN)           /* watch the JRN entries for all this */
```

If you try to run APYJRNCHG more than once, you will get messages telling you that it is not possible. You cannot apply a PUT operation, when a record already exists.

3.14.3 Securing Data between Transactions

You want to make sure that all database operations for a transaction are completed, or if disrupted all of them are backed out.

AS/400 commit control does this. It requires that the files involved are journaled.

This is the minimum you have to know:

- The CL command STRCMTCTL (start commit control) must be issued at the start of the job.
- The files involved must be opened under commit control. In A.3, “A Simple RPG Program with Commit Control” on page 187 is an RPG program that opens one file under commit control.
- The programmer defines when a transaction is finished by issuing a COMMIT command. A Commit operation code is also available in the programming language he uses.
- The programmer may also decide to roll back a transaction. The command is ROLLBACK, and it is also available as operation code in different AS/400 languages.
- The CL command ENDCMTCTL is issued before the job ends.

- IPL, start of job, and end of job will roll back all uncommited database operations.

MQ can save transaction data on a queue in the same way. The only difference is that the PUT or GET operations must indicate PUT under syncpoint or GET under syncpoint. Also, the job must issue a disconnect from the MQ manager before the ENDCMTCTL.

3.14.3.1 Example

Type the program PGMA (see A.3, “A Simple RPG Program with Commit Control” on page 187) into a member in the source file MYAPP/SRC and compile it into the library MYAPP. You can play around with commit and rollback if you enter the commands from a command line. These commit and rollback operations are of course part of the application program in normal life.

- STRCMTCTL /* start commit control */
- DSPPFM FILE(MYAPP/FILA) /* inspect the file FILA */
- CALL MYAPP/PGMA /* add 3 records to the file */
- DSPPFM FILE(MYAPP/FILA) /* look - 3 new records */
- ROLLBACK /* regret the addings */
- DSPPFM FILE(MYAPP/FILA) /* look - all 3 gone */
- CALL MYAPP/PGMA /* add 3 records */
- CALL MYAPP/PGMA /* add another 3 records */
- DSPPFM FILE(MYAPP/FILA) /* look - 6 new records */
- COMMIT /* commit the last transaction */
- DSPPFM FILE(MYAPP/FILA) /* yes they are still there */

You may also try to add records and sign off from the job, then sign on again and observe that uncommited records are rolled back by the operating system.

3.14.4 Journal Receiver Management

Journal receivers will grow until all disk space is used up, if you don’t manage them.

A practical way to handle receiver management is to create and attach a new receiver every day just before a backup of the files it is serving. You do it by issuing the command:

```
CHGJRN JRN(MYJRN/JRN) JRNRCV(*GEN)
```

This command creates a new receiver, attaches it and detaches the old one. When the file is saved an entry is made in the new receiver.

You may back up the receivers on tape and then delete them. After the file is backed up you can delete the receivers from before the file backup, as they are not necessary any more. A simple set of commands shows this:

```
CHGJRN JRN(MYJRN/JRN) JRNRCV(*GEN)
SAVLIB LIB(MYAPP) DEV(TAP01)
SAVLIB LIB(MYJRN) DEV(TAP01)
DLTJRNRCV JRNRCV(MYJRN/RCV)
```

The success is based on the fact that an attached receiver cannot be deleted (if programmed in a CL program you must monitor for such messages).

Chapter 4. Installation and Migration

Installing and migrating MQSeries for AS/400 is described in Chapter 2 of the *MQSeries for AS/400 Administration Guide*, GC33-1956-00.

In this chapter, we will go into more detail and explain how to apply fixes or PTFs (Program Temporary Fix).

Migrating from earlier releases is described, in detail, in Chapter 2 of the Administration Guide. Therefore, we will only give an overview here.

Installing and migrating requires the AS/400 user profile QSECOFR (security officer), or a user profile with his authority.

4.1 Considerations Before the Installation

Before you begin with the installation you must consider the settings of the following system values:

QCCSID	Coded character set identifier
QLANGID	Language identifier
QUTOFFSET	Coordinated universal time offset
QSYSLIBL	System part of the library list
QUSRLIBL	User part of the library list
QALWOBJRST	Allow object restore option

4.1.1 Coded Character Set Identifier (QCCSID)

This system value has no effect on the installation process, but consider it now. QCCSID determines the CCSID for the job if not overridden by the value in the user profile for the job. When you create MQ Manager (after License Program installation) the CCSID for the job will determine the CCSID for MQ Manager. You cannot specify a CCSID in the CRTMQM command.

If you operate in languages spoken outside North America, you must pay attention to this system value. However, you should not change it without careful planning.

The encoding (CCSID) has a major effect on how the AS/400 will translate data between systems in different countries and between AS/400 and ASCII/ANSI systems (PC and UNIX).

© Copyright IBM Corp. 1998

The redbook *Speak the Right Language with Your AS/400 System*, SG24-2154 is necessary to understand all aspects.

In short, all objects that carry text data in an AS/400 have a language encoding (CCSID), for example, USA has 37 and Sweden has 278. CCSID 65535 (called *HEX), means no conversion between different language encoding will take place.

The CCSID values 0 and 65534 have special meanings.

4.1.2 Language Identifier (QLANGID)

The language ID QLANGID determines the LANGID for a job, if not overridden by the user profile. LANGID gives a default CCSID. This default CCSID will be used if CCSID for the job is 65635 (*HEX). LANGID *ENU* will give a default CCSID 37, and LANGID *SVE* will give a default CCSID of 278.

The reason for QCNTRYID is to make it possible to enforce language encoding despite a QCCSID of 65535.

4.1.3 Coordinated Universal Time Offset (QUTOFFSET)

If your communications environment includes systems in different time zones you should indicate a value that reflects the difference. QUTOFFSET is the offset in hours, that the system has from GMT (Greenwich Mean Time). If it is zero, MQSeries will assume that you are at GMT.

MQSeries does the time stamps in GMT (also called universal coordinated time UTC).

4.1.4 System Part of the Library List (QSYSLIBL)

The library QSYS2 must be in the QSYSLIBL. You should never include any libraries for MQSeries (or other application libraries) in QSYSLIBL.

4.1.5 User Part of the Library List (QUSRLIBL)

All jobs in the system will get this part of the library list, if it not overridden by a value in a *job description* indicated by the user profile.

MQSeries for AS/400 has a *quiesce* program that ends all jobs with the library QMQM in the library list for the job. QMQM should never be included in the system value QUSRLIBL.

It is generally a good practice to let the job decide which application libraries are to be included. So keep the QUSRLIBL as QGPL QTEMP.

4.1.6 Allow Object Restore Option (QALWOBJRST)

The default for this system value is *ALL. If you have changed this because of security concerns, you will have to set it to *ALL or include QMQMDATA and QMQMPROC just during the installation of MQSeries for AS/400.

4.2 Installation

Verify with the packing list that your version of MQSeries for AS/400 has the same primary language (2924 US, 2937 Swedish) as the primary language installed in your system.

Don't get this wrong. The actual language on the panels for 5769-MQ1 V4R2 is still English, but the code is packed together with the primary language code, as if it was translated.

Installing an MQSeries for AS/400 product with the primary language *US English* on an OS/400 with primary language *Swedish* will not give a correct result.

Verify also that there is no earlier version of MQSeries for AS/400 installed.

Signon as QSECOFR (or equal).

Display software resources with the following command:

```
DSPSFWRSC
```

Toggle the view with F11 and scroll forward and back until you are sure that you find no older version of MQSeries for AS/400, such as 5716MQ1 V3R7.

Figure 49 on page 86 shows a Swedish AS/400 panel. No, this panel is not translated. Feature 2937 indicates that Swedish is installed as primary language in library QSYS, while Feature 2924 indicates that US English is installed as primary language in library QSYS2724.

If you find a previous version of MQSeries for AS/400, you must decide on the migration path before continuing the installation. Refer to 4.7, "Migrating to V4R2" on page 100 on how to save your existing MQSeries definitions before installing the new version.

```
                          Display Software Resources
                                                                System:   SEST0002
Resource                     Feature
   ID      Option  Feature    Type     Library     Release
5769999    *BASE    5050      *CODE    QSYS        V4R2M0 L00
5769SS1    *BASE    5050      *CODE    QSYS        V4R2M0 L00
5769SS1    *BASE    2924      *LNG     QSYS2924    V4R2M0 L00
5769SS1    *BASE    2937      *LNG     QSYS        V4R2M0 L00
5769SS1    1        5050      *CODE    QSYS2       V4R2M0
5769SS1    1        2924      *LNG     QSYS2924    V4R2M0
5769SS1    1        2937      *LNG     QSYS2       V4R2M0
5769SS1    2        5050      *CODE    QHLPSYS     V4R2M0
5769SS1    2        2924      *LNG     QSYS2924    V4R2M0
5769SS1    2        2937      *LNG     QHLPSYS     V4R2M0
5769SS1    3        5050      *CODE    QSYSDIR     V4R2M0
5769SS1    3        2924      *LNG     QSYS2924    V4R2M0
5769SS1    3        2937      *LNG     QSYSDIR     V4R2M0
5769SS1    4        5050      *CODE    QMGU        V4R2M0
                                                                          Forts
Press Enter to continue.

F3=Exit   F11=Display descriptions   F12=Cancel   F19=Display trademarks
```

Figure 49. Display Software Resources

To install the product follow these steps:

Step 1. Type GO LICPGM on the command line.

Step 2. Select option 11 (install licensed program).

Step 3. Select the options (see below) of MQSeries for AS/400, that you want to install. You may toggle the view with F11 before typing your selection, see Figure 50 on page 87 and Figure 51 on page 87.

When a product is installed it has *installed status* *COMPATIBLE, as shown in Figure 50 on page 87.

The three parts of MQSeries for AS/400 that you may install are:

***BASE** Mandatory base code

1 Optional, samples in QMQMSAMP

2 Optional, Admin Application in STRMQMADM

To verify that your installation is correct type:

GO LICPGM

Then select option 10 and verify that 5769MQ1 has the status *COMPATIBLE.

```
                          Install Licensed Programs
                                                              System:   RALYAS4C
Type options, press Enter.
  1=Install

       Licensed  Installed
Option Program   Status       Description
  _    5769FN1                AFP DBCS Fonts - Korean
  _    5769FN1                AFP DBCS Fonts - Traditional Chinese
  _    5769FN1                AFP DBCS Fonts - Simplified Chinese
  _    5769FN1                AFP DBCS Fonts - Thai
  _    5769FW1                Firewall for AS/400
  _    5763JC1   *COMPATIBLE  AS/400 Toolbox for Java
  _    5769JS1                Job Scheduler for AS/400
  _    5769JV1   *COMPATIBLE  AS/400 Developer Kit for Java
  _    5769MG1   *COMPATIBLE  Managed System Services for AS/400
  1    5769MQ1                MQSeries for AS/400
  1    5769MQ1                MQSeries for AS/400 - Samples
  1    5769MQ1                MQSeries for AS/400 - Admin Application
  _    5769NCE   *COMPATIBLE  Internet Connection Secure Server (Int'l)
                                                                        More...
F3=Exit   F11=Display release   F12=Cancel   F19=Display trademarks
```

Figure 50. Install Licensed Programs - View 1

```
                          Install Licensed Programs
                                                              System:   RALYAS4C
Type options, press Enter.
  1=Install

       Licensed  Product
Option Program   Option   Description
  _    5769FN1   2        AFP DBCS Fonts - Korean
  _    5769FN1   3        AFP DBCS Fonts - Traditional Chinese
  _    5769FN1   4        AFP DBCS Fonts - Simplified Chinese
  _    5769FN1   5        AFP DBCS Fonts - Thai
  _    5769FW1   *BASE    Firewall for AS/400
  _    5763JC1   *BASE    AS/400 Toolbox for Java
  _    5769JS1   *BASE    Job Scheduler for AS/400
  _    5769JV1   *BASE    AS/400 Developer Kit for Java
  _    5769MG1   *BASE    Managed System Services for AS/400
  1    5769MQ1   *BASE    MQSeries for AS/400
  1    5769MQ1   1        MQSeries for AS/400 - Samples
  1    5769MQ1   2        MQSeries for AS/400 - Admin Application
  _    5769NCE   *BASE    Internet Connection Secure Server (Int'l)
                                                                        More...
F3=Exit   F11=Display status   F12=Cancel   F19=Display trademarks
```

Figure 51. Install Licensed Programs - View 2

If the product did not install correctly check the job log by typing DSPJOBLOG on the command line, F10 and scroll back. You will also get more information if you execute the command *Check Product Option*:

```
CHKPRDOPT PRDID(5769MQ1)
```

If you don't get the message No errors detected by CHKPRDOPT then check your job log, to see what went wrong.

4.3 PTFs (Program Temporary Fixes)

After the installation completes successfully, you must install the latest PTFs. The following internet page has the latest information:

```
http://as400service.rochester.ibm.com/
```

- Select **Technical Information Database**.
- Select **Preventive Service Planning (PSP)**.
- Select **All Preventive Service Planning Documents by Release**.
- Scroll the items under R420.
- Select **Fix summary listing for Version 4 Release 2.0.**
- Scroll down to the information about MQSeries for AS/400 .

Figure 52 on page 89 shows the list from May the 4th 1998.

PKG# 8041 indicates that the PTF (fix) is available in the cumulative fix package (a CD) referred to as TC98041 (produced 1998 day 41).

To see the latest fix package that has been installed on your system type the command:

```
DSPPTF LICPGM(5769SS1)
```

Figure 53 on page 90 shows the first of the panels that displays the PTFs applied.

Although the latest cumulative fix package has been applied to the system, the fixes cannot be applied to a product before it is installed. If 5769-MQ1 has been installed as above, the fixes, obviously, have not been applied. To load and apply them follow these steps:

- Put the latest fix package in the CD-ROM drive and type the command:

  ```
  LODPTF LICPGM(5769MQ1) DEV(OPT01)
  ```

- To apply the fixes type the command:

  ```
  APYPTF LICPGM(5769MQ1)
  ```

You should see the list of PTFs shown in Figure 52 on page 89.

```
            PRODUCT NAME: 5769MQ1 - V4 R2.0 - MQSERIES FOR OS/400

|PTF/     |       | AVAIL.  |                                                |REPLACED
|FIX #    |PKG#   |  DATE   |ABSTRACT                                        |   BY
-------------------------------------------------------------------------------
SF43258  8028  01/31/98 MQM400-MSGMCH3401-MSGRC20806058 AMQIQEM4  SF44198
                        DELETES QXSMEMTOP
SF43307  8028  01/31/98 MQM400 IMPROVE MIGRATION OF THE           SF44856
                        SYNCHRONIZATION FILE
                        MQM400 MIGRATION OF 3.6 CHANNEL
                        DEFINTION FILE
SF43461  8028  01/31/98 MQM400 MIGRATION OF MQSERIES CHANNEL      SF44856
                        DEFINITION FILE
SF44104  8041  02/16/98 MQM400 SERVICE UPDATES TO MQSERIES CPP
SF44198  8041  02/16/98 MQM400 UPDATE PROGRAM AMQIQEU4, SHOULD
                        BE DOMAIN *USER
                        MQM400-MSGCPF1321 QUIESCE ENDJOB
                        AMQALMP4 FAILS AFTER ENDMQM
                        MQM400 SERVICE UPDATES TO MQSERIES MCS
SF44856  8041  02/16/98 MQM400 PROGRAM CHECK IN MCA RECEIVER JOB
                        MQM400 SERVICE UPDATES TO MQSERIES MCA &
                        MIGRATE TOOLS
SF45237  8041  02/16/98 MQM400 SERVICE UPDATES TO MQ COMMON       SF45925
                        SERVICES_(MCS)
                        MQM400-MSGAMQ8101-MSGAMQ8137 STRMQM
                        FAILS ALREADY STARTING
                        MQM400 CPC1125 ENDJOB STORAGE MONITOR
                        AMQXIHK4 MCH3601 EXCEP
SF45304  8041  02/16/98 MQM400 - MSGAMQ7002 RC11904603            SF45925
                        LRCE_S_FILE_ERROR AND MSGMCH3
                        MQM400 SERVICE UPDATES TO MQSERIES MQ
                        KERNAL LQM
SF45925  NONE  05/04/98 MQM400-MSGAMQ0825 RC11404704 TEXT NOT
                        AVAIL
                        MQM400-MSGCPF1266 ON STRMQM WHEN QMQM
                        NOT AUTH TO USER LIBS
                        MQM400-UNPRED GETS AND PUTS TO MESSAGE
                        QUEUE DOES NOT UPDATE
                        MQM400 4P20 SHIP COMPONENT XCS PARTS FOR
                        SA69075
                        MQM400 SERVICE - ADIRESIZE RC11503700
                        KEEP MCH2601 MSG DATA
                        MQM400-MSGAMQ6150 JOB AMQALMP4 AND
                        ANOTHER MQSERIES JOB HANG
                        MQM400-MSGCPD0084 ISSUED DURING STRMQM
                        WITH CCSID(5026)
```

Figure 52. PTF Listing

```
                              Display PTF Status
                                                                System:   RALYAS4C
Product ID  . . . . . . . . . . . . . :   5769SS1
IPL source  . . . . . . . . . . . . . :   ##MACH#B
Release of base option  . . . . . . . :   V4R2M0 L00


Type options, press Enter.
  5=Display PTF details   6=Print cover letter   8=Display cover letter

     PTF                                                        IPL
Opt  ID       Status                                          Action
 _   TC98045  Temporarily applied                             None
 _   TC98041  Superseded                                      None
 _   TC98028  Superseded                                      None
 _   TA98045  Temporarily applied                             None
 _   TA98041  Superseded                                      None
 _   TA98028  Superseded                                      None
 _   SF47648  Temporarily applied                             None
 _   SF47628  Temporarily applied                             None
 _   SF46610  Temporarily applied                             None
                                                                          More...
F3=Exit   F11=Display alternate view   F17=Position to   F12=Cancel
```

Figure 53. Display PTF Status

The APYPTF command will apply the PTFs *temporarily*. When the fixes are applied temporarily, they can be removed, should they prove later to be no good.

The PTF SF45925 was not available in a cumulative package (as of May 5, 1998). It must be ordered via the AS/400 *ECS line*. Your AS/400 operator can do this with the following command:

```
SNDPTFORD PTFID(SF46925)
```

Let us use the fix SF46925 for 5769MQ1 as an example. After you have received the PTF and the cover letter via the modem from your ECS line, you must load and apply the PTF with the following commands:

```
LODPTF LICPGM(5769MQ1) DEV(*SERVICE) SELECT(SF45925)
APYPTF LICPGM(5769MQ1) SELECT(SF46925)
```

Don't forget to read the cover letter. Some PTFs require more actions than load and apply.

Cover letters are also available on the internet. The URL is:

```
http://as400service.rochester.ibm.com/
```

To find the cover letters:

- Select **Technical Information Database**.
- Select **AS/400 PTF Cover Letters**.
- Click on **Search**.
- Type 5769MQ1 in the search field and click on **Search**.
- Select and read the current cover letters.

4.4 Verifying the Installation of 5769MQ1

Check the record length of the following files:

- The channel definition file QMQMDATA/AMQRFCD4
- The channel synchronization file QMQMDATA/AMQRSYNA

Unfortunately, early MQ V4R2 code has been distributed with a wrong record length for AMQRFCD4.

The correct record lengths are:

- AMQRFCD4 - 2720
- AMQRSYNA - 24444

If this is not the case, check that you have applied all PTFs and followed the instructions in the cover letters. You will have to re-create the files, if they are wrong, with the following commands:

```
DLTF QMQMDATA/AMQRSYNA
CRTPF FILE(QMQMDATA/AMQRSYNA) SRCFILE(QMQM/QDDSSRC)
CHGOBJOWN OBJ(QMQMDATA/AMQRSYNA) OBJTYPE(*FILE) NEWOWN(QMQM)

DLTF QMQMDATA/AMQRFCD4
CRTPF FILE(QMQMDATA/AMQRFCD4) SRCFILE(QMQM/QDDSSRC)
CHGOBJOWN OBJ(QMQMDATA/AMQRFCD4) OBJTYPE(*FILE) NEWOWN(QMQM)
```

After the installation the libraries shown in Table 6 on page 92 should exist.

The installation creates message files with descriptions of MQSeries messages (error messages, confirmation messages, etc.) in the operator and job log in QSYS and QSYSxxxx:

- QSYS/AMQMSG
- QSYSxxxx/AMQMSG - (QSYS2924 US English)

Table 6. Files Changed in V4R2

Library	Install Option	Content
QMQM	*BASE	• Commands • Programs • Message layout for programmers
QMQMDATA	*BASE	• Journal receivers • Channel files • MQ objects: queues and channels
QMQMPROC	*BASE	• Empty from start • Defined processes
QMQMSAMP	1 (Samples)	• Source files
QMQMADM	2 (Admin)	• Commands • Programs

After the installation MQM commands exists in the following libraries:

QMQM	MQSeries for AS/400
QSYS	System Library for AS/400 (OS/400)
QSYSxxxx	Secondary Language Library (QSYS2924 US English)
QSYSV3R2M0	Previous Version Library V3R2
QSYSV3R7M0	Previous Version Library V3R7
QSYSV4R1M0	Previous Version Library V4R1M0
QSYSV4R1M4	Previous Version Library V4R1M4

Only the MQM commands end up in the QSYS-type libraries. The actual programs behind the commands remain in the QMQM library.

If you plan to change the standard authority for accessing MQ commands, keep all these libraries in mind. Usually, the QSYS commands are executed, because QSYS is at the beginning of the system part of the library list.

You do not need to have QMQM in the library list in order to work with MQ commands.

The journals for managing persistence in MQSeries are created when you create the queue manager, not when you install the product code.

4.5 Creating the Queue Manager

In an AS/400, you can have only one MQSeries queue manager. Therefore, you may just as well create it now as the first step after the installation.

You need to be a security officer (QSECOFR) to do that. Sign on as QSECOFR or as MQ adminstrator (MQADM). How to create an MQ administrator is described in Chapter 5, “Security Considerations” on page 107.

4.5.1 Choosing the CCSID

Before creating the MQ manager, display the CCSID (Coded Character Set Identifier) of the job using the following command:

```
DSPJOB OPTION(*DFNA)
```

Then scroll forward to the last panel, shown in Figure 54.

```
                     Display Job Definition Attributes
                                                          System:   SESTOCK8
Job:   QPADEV000E     User:   KLAUS          Number:   012102

Print key format  . . . . . . . . . . . . . . . . . :   *PRTHDR
Sort sequence . . . . . . . . . . . . . . . . . . . :   *LANGIDSHR
  Library . . . . . . . . . . . . . . . . . . . . . :
Language identifier . . . . . . . . . . . . . . . . :   SVE
Country identifier  . . . . . . . . . . . . . . . . :   SE
Coded character set identifier  . . . . . . . . . . :   278
Default coded character set identifier  . . . . . . :   278
Job message queue maximum size  . . . . . . . . . . :   16
Job message queue full action . . . . . . . . . . . :   *NOWRAP
Allow multiple threads  . . . . . . . . . . . . . . :   *NO









                                                                       Bottom
Press Enter to continue.

F3=Exit   F5=Refresh   F12=Cancel
```

Figure 54. Display Job Definition Attributes

If the CCSID (bold in Figure 54) has a value other than 65535 (*HEX), this value will be the CCSID for the MQ manager. If the CCSID for the job is 65535, the *language identifier* (LANGID) will determine the CCSID for the queue manager. The language ID SVE results in CCSID 278 (Swedish), while the language ID 37 results in CCSID 37 (US English).

To determine the CCSID of your choice for the queue manager you may change the CCSID or LANGID before you create it by typing:

```
CHGJOB LANGID(SVE) CCSID(278)
```

4.5.2 What Is the CCSID for My Language?

You can find out the CCSID of your language ID by setting the CCSID for your online job to *HEX. Then set the language ID, and look at the resulting default CCSID. That is your CCSID. Do this:

1. Enter the command DSPJOB OPTION(*DFNA) and scroll to the bottom of the Job Definition Attributes panel. There you find a language ID and CCSID.
2. Change the CCSID with CHGJOB CCSID(65535).

```
                  Specify Value for Parameter LANGID

Type choice, press Enter.

Type . . . . . :   CHARACTER
Language ID  . . . . . .   ENU___

*SAME                              CAT Catalan
*SYSVAL                            HRV Croatian
*USRPRF                            CSY Czech
AFR Afrikaans                      DAN Danish
SQI Albanian                       NLD Dutch
ARA Arabic                         ENA English Australian
NLB Belgian Dutch                  ENP English Upper Case
FRB Belgian French                 EST Estonian
ENB Belgium English                FAR Farsi
PTB Brazilian Portuguese           FIN Finnish
BGR Bulgarian                      FRA French
BEL Byelorussian                   DEU German
FRC Canadian French                ELL Greek                              +

F3=Exit   F5=Refresh   F12=Cancel   F13=How to use this display   F24=More keys
```

Figure 55. List of Language IDs (LANGID)

3. A list of language IDs is provided in the panel shown in Figure 55. To display the panel follow these steps:
   a. Type the command CHGJOB without parameters and do *not* press Enter.
   b. Press the following keys in this sequence: F4, F10, F11 and scroll to the bottom of the panel.
   c. In the field LANGID type a ?, press the spacebar and then Enter.

d. Now you see the panel shown in Figure 55.

4. Scroll through the help panel to find the appropriate language ID. Type that code (BEL for Byelorussian, for example) in the input field for LANGID and press Enter.
5. Execute the command DSPJOB OPTION(*DFNA) again and scroll to the end of the panel. Watch the default CCSID. When CCSID is *HEX (65535), default CCSID reflects the CCSID corresponding to LANGID (1025 - BEL).
6. Use CHGJOB to change LANGID and CCSID back to your choice.

4.5.3 Choosing a Queue Manager Name

Select a *unique name* in your network for the MQ manager. We recommend that you choose the *system name*.

The system name can be displayed with the command:

```
DSPNETA
```

Don't use the serial number for the system. That makes life difficult, when you replace the system with another one some day.

This name will be known to other systems that use case sensitive names.

If you key the name without blank characters in the name field, and without single quotes (') around, it will be in uppercase. Otherwise, you will get what you key, mixed upper and lowercase, as shown in the following examples:

Table 7. Case-Sensitive MQMNAME

Name keyed	Result
as400.as400ral4.Raleigh	AS400.AS400RAL4.RALEIGH
'as400.as400ral4.Raleigh'	as400.as400ral4.Raleigh
as400 raleigh	as400 raleigh

Using blanks can be a burden.

4.5.4 The CRTMQM Panel

With the following command bring up the Create Message Queue manager panel:

```
crtmqm
```

Do *not* press Enter, but press F4 instead. This displays the panel as shown in Figure 56 on page 96.

```
                    Create Message Queue Manager (CRTMQM)

Type choices, press Enter.

Message Queue Manager name . . .   ralyas4c_____________________________________
_____
Text 'description' . . . . . .   MQM for ralyas4c_____________________________
_____________________________
Trigger interval . . . . . .   999999999__   0-999999999
Undelivered message queue  . . .   System.dead.letter.queue_____________________
_____
Default transmission queue . . .   *NONE_____________________________________
_____
Maximum handle limit . . . . . .   256________   0-999999999
Maximum uncommitted messages . .   10000______   1-10000




                                                                        Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 56. Create Message Queue Manager

```
                    Create Message Queue Manager (CRTMQM)

Type choices, press Enter.

Message Queue Manager name . . . MQMNAME      > RALYAS4C_____________
_____________
Text 'description' . . . . . . . TEXT        > 'mqm for ralyas4c'__
_____________________________________
Trigger interval . . . . . . . . TRGITV        999999999__
Undelivered message queue  . . . UDLMSGQ      > SYSTEM.DEAD.LETTER.QUEUE_______
_____________
Default transmission queue . . . DFTTMQ         *NONE_____________________________
_____________________
Maximum handle limit . . . . . . MAXHDL         256________
Maximum uncommitted messages . . MAXUMSG        10000______




                                                                        Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 57. Create Message Queue Manager (After F11)

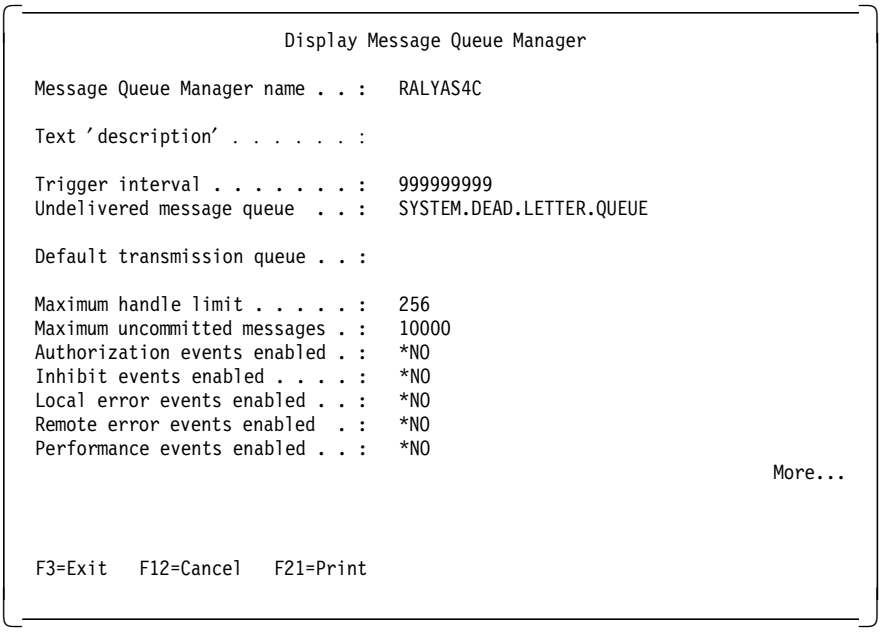


Figure 58. Display Message Queue Manager

Enter the queue manager name, here RALYAS4C, and a desription. When you create the MQ manager, specify also a queue name for undelivered messages. The input is shown in **bold**.

Then press F11 to watch how AS/400 interprets your input. Figure 57 on page 96 shows the names in uppercase. This panel also shows the keyword for the parameters (for example, MQMNAME, TEXT).

Note: You may also add or change the undelivered message queue later with the CHGMQM command.

4.5.5 Creating and Starting the MQM

Press Enter and the MQ manager will be created. You can display its attributes with the DSPMQM command. This command requires that the queue manager is running. To start the MQM and display its attributes issue the following commands:

```
STRMQM
DSPMQM
```

You see the first part of the attributes in Figure 58.

Press PgDn to scroll to the next panel, as shown in Figure 59 on page 98, which displays the rest of the attributes.

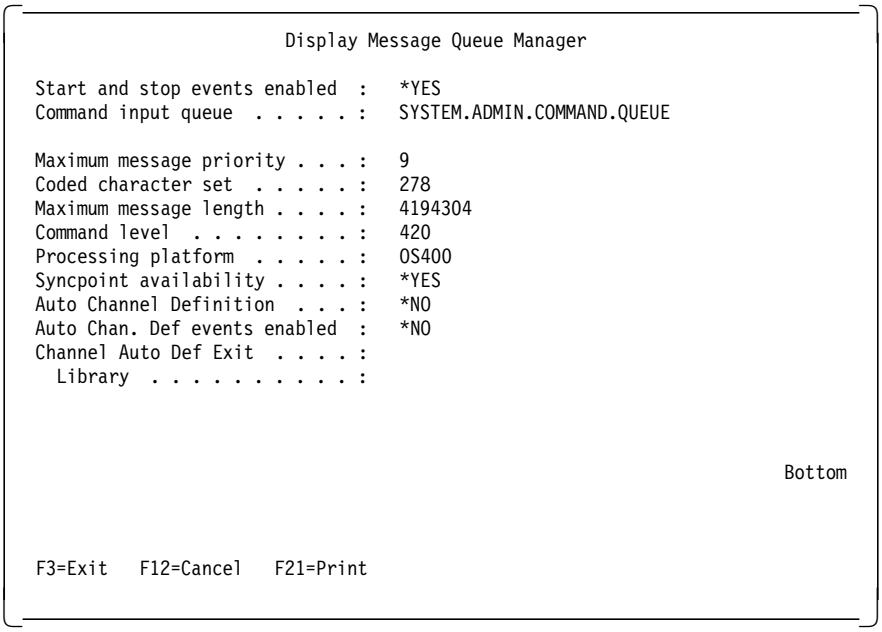


Figure 59. Display Message Queue Manager

4.5.6 Creating the Default Environment

Each queue manager needs a set of objects to work with. These objects, channels, queues and processes are described in 2.1.3, “MQSeries Objects” on page 11. One of the objects, the SYSTEM.DEAD.LETTER.QUEUE, has already been discussed. You create the default objects with the following command:

```
CALL QMQM/AMQSDEF4
```

4.5.6.1 Did You Get an Error?

If you got an error, check that the file QMQMDATA/AMQRFCD4 has the record length 2720 (for V4R2). If the length is wrong, re-create the file as described in 4.4, “Verifying the Installation of 5769MQ1” on page 91 and run the standard definitions again.

You can create the very same standard definitions by executing an MQ command script file instead of calling AMQSDEF4:

```
STRMQMMQSC SRCMBR(AMQSCOMA) SRCFILE(QMQMSAMP/QMQSC) OPTION(*VERIFY)
STRMQMMQSC SRCMBR(AMQSCOMA) SRCFILE(QMQMSAMP/QMQSC) OPTION(*RUN)
```

If you use this approach be sure to correct the syntax errors for definitions 11 and 12 (SYSTEM.DEF.SVRCONN and SYSTEM.AUTO.SVRCONN), that have

been distributed with the early Version 4 Release 2. The syntax error is an NPMSPEED specification, and it should be left out. NPMSPEED is not allowed for MQI channels.

4.5.6.2 Journal and Receivers

After the MQM is created two journals are placed in the library QUSRSYS:

- QUSRSYS/AMQAJRN - MQM local journal
- QUSRSYS/AMQRJRN - MQM remote journal

The corresponding journal receivers are created in QMQMDATA:

- QMQMDATA/AMQA000000 - MQM local journal receiver
- QMQMDATA/AMQR000000 - MQM remote journal receiver

4.5.6.3 Where to Find the Definitions

The source code is listed in Appendix A of the *MQSeries for AS/400 Administration Guide V4R2*, GC33-1956.

You can view the source code for the default definitions online. They are in the member AMQSDEF4 in the source file QMQMSAMP/QCLSRC. To look at them use the AS/400 Program Development Manager (PDM) and follow these steps:

1. Type STRPDM on the command line.
2. Select 1, work with libraries.
3. Enter QMQM* in the Library field (don't forget the asterisk) and press Enter.
4. Key 12 (option work with) in front of QMQMSAMP and press Enter.
5. Scroll forward and type 12 (work with) in front of the source file name QCLSRC.
6. Scroll and key 5 (display) in front of the source code AMQSDEF4.

Remember: Shift+F11 shows more options.

4.5.7 Deleting the Queue Manager

If you want to change the name of the queue manager, you have to delete the existing one after first stopping it. Refer to the following commands as an example:

```
ENDMQM MQMNAME(RALYAS4C)
DLTMQM MQMNAME(RALYAS4C)
CRTMQM MQMNAME(TEST1)
STRMQM MQMNAME(TEST1)
DSPMQM MQMNAME(TEST1)
```

After the DLTMQM command has been executed most MQM definitions are gone. They were in the MQ catalog, with the same name as the MQM. They are:

- MQM manager
- MQ queue definitions
- MQ channel definitions
- MQ process definitions
- MQ local journal and local journal receiver

4.6 Client Software for Clients Connected to the AS/400

Client software is not distributed with the CD-ROM containing MQSeries for AS/400.

The client software is available for free on the Internet. Download it from the Transaction Processing SupportPacs library using the URL:

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

4.7 Migrating to V4R2

This section contains an overview of the migration from previous versions of MQSeries for AS/400. Refer to the *Administration Guide V4R2* for more details.

It is important that you know the following:

- In V4R2, new code has been added to QMQM.
- With V4R2, new *Version 2* structures have been introduced, for example, the message header, as well as the put and get message options. They allow programmers to use the new features of V4R2. The old Version 1 structures are still available.
- The channel description and synchronization files in QMQMDATA have changed formats, so old V3R7 files describing and synchronizing channels cannot be used.

The key to a successful migration is:

- All definitions and changes to the old MQ installation should be available in CL programs or command lists.
- The new files must have the new format.

The administration manual describes three migration scenarios:

1. Product code only update
2. Product code and definitions upgrade
3. Product code, definitions and message data upgrade

The first two scenarios require that the base option of the old MQSeries is deleted. To be practical, delete all three options:

*BASE Mandatory base code

1 Optional, samples in QMQMSAMP

2 Optional, Admin Application in STRMQMADM

Usually, mixing different versions of the same licensed code causes problems.

If you delete the base option of MQSeries for AS/400, you delete the following libraries:

- QMQM
- QMQMDATA
- QMQMPROC

In addition, the QM commands will be removed from the system libraries. In other words: *Keep all other application objects out of these libraries*.

Check that all objects beginning with AMQ have really been deleted from the QSYS, QSYSxxxx and QUSRSYS libraries.

You may have had MQ jobs running during the delete without knowing it. Objects locked during execution cannot be deleted. That is why you are advised to *quiesce* the MQ manager before the migration. Objects surviving from earlier releases may produce very strange errors.

4.7.1 Preparing for Migration

Before migrating you should take a *tape backup* of the objects that you want to continue working with after the migration. You should always do that before you apply major changes, because this is not a task you perform frequently and things could go wrong by mistake.

4.7.1.1 Stop MQSeries

Before the backup you must bring the queue manager, all MQ jobs, and related applications to a complete stop. This means:

- Stopping people from running MQ applications
- Stopping people from signing on MQ applications
- Stopping applications using MQ in a controlled way
- Stopping MQ channels
- Stopping MQ listener
- Stopping MQ command server
- Stopping MQ manager
- Changing journal receiver (generic)
- Restoring media recovery point
- Stopping all other jobs using MQ (quiescing)
- Cleaning up MQ cache storage

Note: For more information refer to:

8.3, “Starting MQ Operations” on page 162

8.4, “Stopping MQ Operations” on page 163

If you are moving to a new machine, planning this is a separate task outside the scope of this book. For MQ it is essential that configuration objects and security objects are saved and restored to the new site.

4.7.1.2 What to Include in a Backup

You do not expect the migration to fail. However, you naturally plan for that possibility which ultimately means reinstalling the previous version and restoring the old MQ objects. Backups should include:

- Application programs and source
- Application files
- MQ libraries:
  - SAVLICPGM and SAVLIB of QMQM
  - QMQMDATA
  - QMQMPROC
  - QMQMSAMP
  - QMQMADM

  You may have forgotten that you have stored user data in some of these libraries.

- A more selective backup of the CL programs that can re-create your MQ objects, similar to the program QMQM/AMQSDEF4
- A text file with MQ commands that can re-create your MQ objects, similar to the member AMQSCOMA in QMQMSAMP/MQSC
- The channel synchronization file QMQMDATA/AMQRSYNA
- The channel definition file QMQMDATA/AMQRFCD4
- The file MQMQDATA/QMINI (if you have changed TCP/IP ports)
- The journals QUSRSYS/AMQAJRN and QUSRSYS/AMQRJRN
- The journal receivers need no backup. Restoring a journal creates a new receiver if a receiver is not present at restore time.

To save your own definitions in a CL command file (to be complied) or in an MQSC file is not necessarily as big of a burden as you might think it is. You will get the standard and sample definitions with the programs in the new product code.

4.7.1.3 Guidelines to Re-create an Environment

The following is a guide to ensure that CL programs really can re-create your existing environment:

- Save the source for the CL program you used for creating the standard definitions QMQMSAMP/QCLSRC(AMQSDEF4).
- Print it.
- Compare the listing with your actual standard definitions (WRKMQMQ, WRKMQMCHL).
- Copy the source of AMQSDEF and correct it to reflect the actual standard definitions, such as SYSTEM.DEF.RECEIVER.
- Forget the sample definitions unless you use them in your applications.
- Create a program that defines your actual queues, channels, and processes, but with new names, such as NEW.XXX.YYY.
- Compile and run the definitions.
- Compare these definitions with your actual ones.
- Change the program to recognize your actual names (XXX.YYY).
- Save it offline.

4.7.2 Update Product Code Only

As it is described in the *Administration Guide*, this is really scrapping the old MQ installation product code including MQ manager definitions and installing the new MQ version product code and creating a new MQ manager, and thus new queues, processes and channe definitions.

1. End all old MQ operations (quiesce).
2. DLTLICPGM of the old MQ product (5716MQ1).
3. Check that all MQ libraries and objects are deleted.
4. GO LIGPGM, type 11 and install the needed options.
5. Apply PTFs.
6. Check the record length of the channel files in QMQMDATA.
7. Check the communications setup (LU 6.2, TCP/IP).
8. Create a new MQ manager, new queues, etc.
9. Check security.
10. Start and initiate MQM operation as normal.
11. Test that MQ objects and MQ applications are working OK.

4.7.3 Upgrade Product Code and Definitions

This is the most straightforward scenario. However, in order to use it you must be able to completely create your MQ setup again.

You can re-create your MQ setup definitions with either CL commands or with standard MQ commands.

- MQM CL commands are compiled into a *CL program*. For example, the program QMQM/AMQSDEF4 creates the standard definitions for the sample programs. See the source in the member AMQSDEF4 in the source file QMQMSAMP/QCLSRC.
- Standard MQ commands (MQSC is the same on all platforms) are in a *source member*. The source member AMQSCOMA in the source file QMQMSAMP/QMQSC is an example.

If these programs are in place, do the following:

1. End all old MQ operations (quiesce).
2. DLTLICPGM of the old MQ product (5716MQ1).
3. Check that all MQ libraries and objects are deleted.
4. GO LIGPGM, type 11 and install the needed options.

5. Apply PTFs.
6. Check the record length of the channel files in QMQMDATA.
7. Compare the new standard definition program with the one you had before the installation (source of AMQSDEF).
8. Create the standard definitions with the command `CALL MQMQ/AMQSDEF4`. Use the program that creates them as they were before. Do not forget the new V4R2 definitions such as SYSTEM.AUTO.RECEIVER.
9. Create your own definitions with similar program(s).
10. Create additional definitions with STRMQSC, if you also have definitions in a text file with MQSC commands.
11. Check security.
12. Check the communications setup (LU 6.2, TCP/IP).
13. Start and initiate MQM operation as normal.
14. Test that MQ objects and MQ applications are working OK.

4.7.4 Upgrade Product Code, Definitions and Message Data

In this scenario you do not delete the old licensed code, so the install program will change the existing libraries QMQM and QMQMDATA.

The success of this scenario is highly dependent on the ability of old MQ objects to work together with the ones of the new release. If you are in doubt, choose the previous scenario.

The two files in Table 8 may give you problems:

Table 8. Files Changed in V4R2

Function	File in QMQMDATA	V3R7 length	V4R2 length	Migrate Programs in QMQM
Channel Synchronization File	AMQRSYNA	24,122	24,444	AMQICLS4 AMQICNS4
Channel Definition File	AMQRFCD4	944	2 720	AMQICNV4

You should check the record length of the files after the installation of the licensed code.

If a file needs to be converted, remember to use the programs that are supplied with the latest PTFs.

The conversion is described in the *Administration Guide* and in the cover letter for the PTFs.

The way the conversion programs work is as follows:

1. Stop journaling of the file.
2. Copy the file to the current library (QGPL) to have a spare.
3. Create an empty file with the new format in QTEMP.
4. Update the QTEMP copy with the data from the QMQMDATA copy.
5. Delete the old file in QMQMDATA.
6. Copy the QTEMP file to QMQMDATA.
7. Delete the QTEMP file.
8. Start journaling the converted file in QMQMDATA.

After this you can find your old file in the current library. Take a backup of it and delete it from the current library, so that future operations are not disturbed.

Unfortunately, some early code did this conversion wrong, so new conversion programs are supplied in new PTFs. The cover letter provides step-by-step instructions.

What follows is a summary of the steps for such a case:

1. GO LIGPGM, type 11 and install the needed options (all).
2. Apply PTFs.
3. Do the conversion of QMQMDATA/AMQSYNA and QMQMDATA/AMQSFCD4 as explained in the PTFs and page 20 in the *Administration Guide*.
4. Check the record length of the channel files in QMQMDATA.
5. Check security.
6. Control the communications setup
7. Start and initiate MQM operation as usual.
8. Test that MQ objects and MQ applications are working properly.

Chapter 5. Security Considerations

MQSeries for AS/400 has implemented security somewhat differently from the standard AS/400 way.

MQSeries uses AS/400 object security when working with normal AS/400 objects, such as commands and programs. However, for queue manager objects, such as queues and channels, MQSeries has implemented its own security. This MQ specific security works against the 48 character long names for MQ objects, such as queue manager, queues and processes.

The commands that control authority for MQ objects are:

DSPMQMAUT	Display MQM Object Authority
GRTMQMAUT	Grant MQM Object Authority
RVKMQMAUT	Revoke MQM Object Authority

Compare this to the well known commands DSPOBJAUT, GRTOBJAUT, RVKOBJAUT and EDTOBJAUT (edit object authority).

An MQM object such as the SYSTEM.DEAD.LETTER.QUEUE is known to the OS/400 operating system as user space (*USRSPC), here QQ0Q000000.

AS/400 objects have short names, a 10-character object name plus a 10-character library name. MQSeries works with object names up to 48 characters long for all platforms but the AS/400.

Don't use the AS/400 standard security commands against the short names for these MQ objects.

This is how the MQ security works (as shipped):

- QSECOFR and all users with *ALLOBJ have the authority *ALL for all MQM objects.
- QMQM (supplied with MQSeries for AS/400) also has the authority *ALL for all MQ objects.
- No one else has access to MQ objects, unless granted with GRTMQMAUT.
- When you execute MQ commands, access to an object is controlled.
- Jobs that call program interfaces (MQI), for example, to put a message on a queue, will be checked when the queue is opened.

© Copyright IBM Corp. 1998

- For subsequent GETs and PUTs authority is not checked. Access is allowed because the API makes the job *adopt* the user profile of QMQM.

5.1 MQ Command Security

The use of MQ CL commands is controlled by standard AS/400 security, since they are AS/400 objects of the type *CMD.

As MQSeries for AS/400 is shipped, every user may use the commands, with the exception of:

- Service and trace commands
- Channel commands

This public access to the MQ commands causes no security risk because you cannot work with MQ objects without being granted authority.

Anyway, you may be annoyed by the fact that everybody has access to the MQ authority commands, such as GRTMQMAUT. To limit the access to these commands, use the standard AS/400 authority commands, such as EDTOBJAUT and RVKOBJAUT.

Just remember, these commands exist in multiple libraries. Some of them are QSYS, QSYS2924, QMQM and QSYSV4R1M4. Refer to page 92 of this document for a list of libraries.

Example:

The following job log shows that you need more than access to the MQ commands in order to use them in a proper way:

```
> CRTMQMQ QNAME(TESTX) QTYPE(*LCL)
  Error found on CRTMQMQ command.
> DSPJOBLOG
  End of requests.
> CRTMQMQ QNAME(TESTX) QTYPE(*LCL)
  MQM queue created.
```

1. With the first command, the user wanted to create the local queue TESTX, but the command failed. Why?
2. To find out why the command failed, display the job log with the command DSPJOBLOG.

   The job log contains more information about what happened when you executed the commands. To browse through the log press F10, then PageUp, and then F1 with the cursor at the message Not authorized.

Figure 60 on page 109 shows the additional message information.

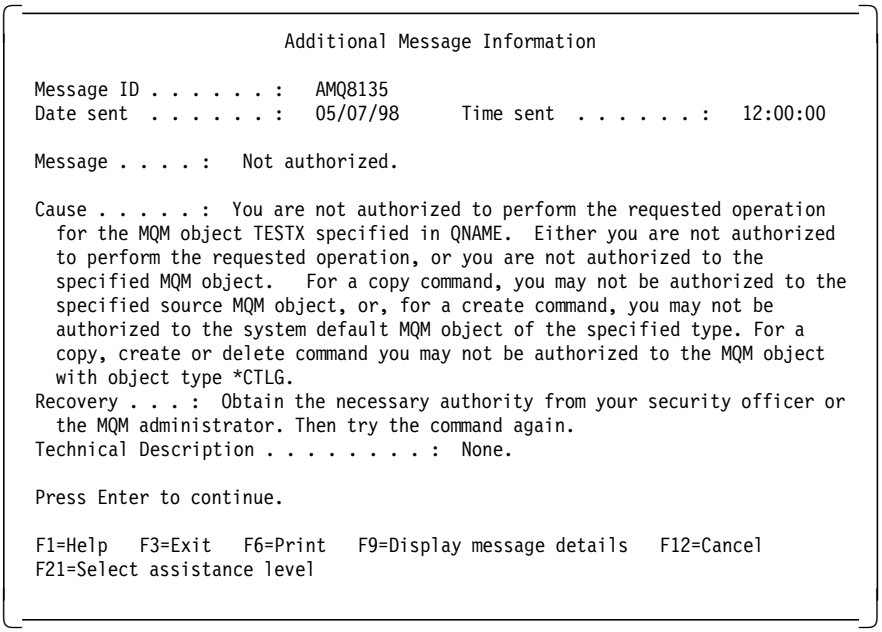


Figure 60. Security - Not Authorized Message

3. After the DSPJOBLOG command, from another screen, QSECOFR has granted the user (MQPGM) access to the MQ manager catalog object **CTLG*. That is why the second create queue operation was successful.

Granting authority is done with the following commands (provided the MQ manager name is MQ400):

```
GRTMQMAUT OBJ(QM400) OBJTYPE(*CTLG) USER(MQPGM) AUT(*USE)
GRTMQMAUT OBJ(QM400) OBJTYPE(*CTLG) USER(MQPGM) AUT(*ADD)
```

The first command allows the user MQPGM to use objects, and the second command authorizes him to add new objects, such as queues.

You can display the object authority for the user with this command:

```
DSPMQMAUT OBJ(QM400) OBJTYPE(*CTLG)
```

Press F11 to see details as shown in Figure 61 on page 110.

```
                        Display MQM Object Authority

Message Queue Manager:   QM400

MQM Object . . . . . :   QM400
  MQM Type . . . . . :     *CTLG

Object . . . . . :   QMQMOBJCAT      Object Type  . . . . :   *USRIDX
  Library  . . . . . :    QMQMDATA     Owner  . . . . . :   QMQM

Object secured by authorization list . . . . . :   *NONE

                       ----Object-----  ----------Data-----------
User        Authority   Opr  Mgt  Exist  Read  Add  Update  Delete
*PUBLIC     *EXCLUDE
QMQM        *ALL         X    X     X     X     X     X       X
MQPGM       USER DEF     X                X     X


                                                                  Bottom
F3=Exit   F12=Cancel   F21=Print
```

Figure 61. Display MQM Object Authority

5.2 MQ User Profiles

QMQM is the only user profile supplied with the product, but it cannot be used to sign on.

If you sign on as security officer (QSECOFR) or as a user with his authority, you can do all tasks ever needed for MQSeries for AS/400 without any authority problems.

This may be OK for installing the product, but usually nobody would allow such a user to do daily work. Not only can this user look into the payroll and change it, he could also easily, by mistake, delete crucial information.

We see a need for at least four types of profiles when working with MQSeries:

1. An administrator, to create and control the MQ environment
2. A programmer, using the programming interface, and creating queues
3. An application user, calling programs that access MQ queues
4. An operator who can start and stop the MQ manager and control channels

You can give a user profile one of five security classes. The classes, ranked by authority are:

**SECOFR*	Security officer
**SECADM*	Security administrator
**PGMR*	Programmer
**SYSOPR*	System operator
**USER*	User

The *Administration Guide* covers security in detail. The following are just examples to illustrate the different profile roles.

In the following, we create the four user profiles mentioned above to work with MQSeries. To create a profile with a command you type on the command line or via the panel shown in Figure 62. The command is:

```
crtusrprf
```

```
                        Create User Profile (CRTUSRPRF)

Type choices, press Enter.

User profile . . . . . .   _________      Name
User password  . . . . . .   *USRPRF       Name, *USRPRF, *NONE
Set password to expired  . . . .   *NO      *NO, *YES
Status . . . . . .   *ENABLED      *ENABLED, *DISABLED
User class . . . . . .   *USER         *USER, *SYSOPR, *PGMR...
Assistance level . . . . . .   *SYSVAL       *SYSVAL, *BASIC, *INTERMED...
Current library  . . . . . .   *CRTDFT       Name, *CRTDFT
Initial program to call  . . . .   *NONE      Name, *NONE
  Library  . . . . . .                       Name, *LIBL, *CURLIB
Initial menu . . . . . .   MAIN          Name, *SIGNOFF
  Library  . . . . . .     *LIBL         Name, *LIBL, *CURLIB
Limit capabilities . . . . . .   *NO        *NO, *PARTIAL, *YES
Text 'description' . . . . . .   *BLANK



                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
Parameter USRPRF required.
```

Figure 62. Create User Profile Panel

5.2.1 The MQ Administrator (MQADM)

You would probably not let a normal user (*USER) use this profile. Depending on which other task MQADM has to do, you will give this profile one of these user classes:

- *SECADM
- *PGMR
- *SYSOPR

The user must also be defined at remote MQ sites (AS/400, Windows NT, etc.) if he or she is going to perform remote MQ administration.

The following commands create the user profile MQADM of the *PGMR class, with a start menu that displays all MQ CL commands (CMDMQM) instead of the main menu shown on page 46. The user will be granted access to all MQ objects, and will be initialized to run the AS/400 MQ administration application. At the first signon, he must change the password.

```
CRTUSRPRF USRPRF(MQADM) PWDEXP(*YES) USRCLS(*PGMR)
          INLMNU(*LIBL/CMDMQM) TEXT('MQ Administrator')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*ALL) USER(MQADM) AUT(*ALL)
CALL PGM(QMQM/AMQSADM4) PARM(MQADM)
```

The administrator must be allowed to use the service and channel commands:

```
CRTAUTL AUTL(MQSRV) TEXT('MQ service commands')
ADDAUTLE AUTL(MQSRV) USER(MQADM) AUT(*USE)

GRTOBJAUT OBJ(ENDMQMSRV) OBJTYPE(*CMD) AUTL(MQSRV)
GRTOBJAUT OBJ(STRMQMSRV) OBJTYPE(*CMD) AUTL(MQSRV)
GRTOBJAUT OBJ(TRCMQM) OBJTYPE(*CMD) AUTL(MQSRV)

CRTAUTL AUTL(MQCHL) TEXT('MQ channel commands')
ADDAUTLE AUTL(MQCHL) USER(MQADM) AUT(*USE)

GRTOBJAUT OBJ(PNGMQMCHL) OBJTYPE(*CMD) AUTL(MQCHL)
GRTOBJAUT OBJ(RSTMQMCHL) OBJTYPE(*CMD) AUTL(MQCHL)
GRTOBJAUT OBJ(RSVMQMCHL) OBJTYPE(*CMD) AUTL(MQCHL)
GRTOBJAUT OBJ(STRMQMCHL) OBJTYPE(*CMD) AUTL(MQCHL)
GRTOBJAUT OBJ(STRMQMCHLI) OBJTYPE(*CMD) AUTL(MQCHL)
GRTOBJAUT OBJ(STRMQMLSR) OBJTYPE(*CMD) AUTL(MQCHL)
GRTOBJAUT OBJ(WRKMQMCHL) OBJTYPE(*CMD) AUTL(MQCHL)
GRTOBJAUT OBJ(WRKMQMCHST) OBJTYPE(*CMD) AUTL(MQCHL)
```

When MQADM signs on the panel shown in Figure 63 on page 113 will appear. This panel is supplied with MQSeries.

```
CMDMQM                        MQSeries Commands

Select one of the following:

  Queue Manager Commands
     1. Change Message Queue Manager                          CHGMQM
     2. Connect MQM                                           CCTMQM
     3. Create Message Queue Manager                          CRTMQM
     4. Delete Message Queue Manager                          DLTMQM
     5. Disconnect MQM                                        DSCMQM
     6. Display Message Queue Manager                         DSPMQM
     7. End Message Queue Manager                             ENDMQM
     8. Start Message Queue Manager                           STRMQM

  Command Server Commands
     9. Display MQM Command Server                            DSPMQMCSVR
    10. End MQM Command Server                                ENDMQMCSVR
    11. Start MQM Command Server                              STRMQMCSVR

                                                                 More...
Selection or Command
===>
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Figure 63. MQSeries Commands

5.2.2 The MQ Programmer (MQPGM)

The MQ programmer is a normal application programmer, but he must also be granted access to the MQ catalog after the MQ administrator has created the MQ manager.

Note: The AS/400 allows only one queue manager instance.

The programmer needs the library QMQM in the user library list, because it contains source definitions necessary when compiling MQ programs.

Further MQ authorities should be granted by MQADM on a need-to-know basis.

The MQPGM should makes copies of the source code of the sample programs. The QMQMSAMP library should not be changed or become part of an application. The library may be changed in the next release.

Example:

We create the library MQPGM for the programmer. This should be done by a security officer (QSECOFR).

```
CRTUSRPRF USRPRF(MQPGM) PWDEXP(*YES) USRCLS(*PGMR)
        CURLIB(MQPGM) INLPGM(QCMD)
        JOBD(MQPGM/MQPGM) OUTQ(MQPGM/MQPGM)
ADDAUTLE AUTL(MQSRV) USER(MQPGM) AUT(*USE)
ADDAUTLE AUTL(MQCHL) USER(MQPGM) AUT(*USE)

CRTLIB LIB(MQPGM) TEXT('MQ programs')
CHGOBJOWN OBJ(MQPGM) OBJTYPE(*LIB) NEWOWN(MQPGM)
CRTJOBD JOBD(MQPGM/MQPGM) TEXT('MQ programmer jobd')
        SYNTAX(30) INLLIBL(QTEMP QMQM QGPL) LOG(4 0 *SECLVL)
        LOGCLPGM(*YES) ALWMLTTHD(*YES)
CRTOUTQ OUTQ(MQPGM/MQPGM) TEXT('Outputq for print')
```

Notes:

1. Do not type all the commands. Refer to 3.4, "Using Commands" on page 49 on how to use prompting with F4, F11 and F10.
2. QSECOFR may prefer to use a CL program to do all this, so that he is prepared when the next programmer arrives.

CRTAUTL and GRTOBJAUT has been done under MQADM authority. Refer to 5.2.1, "The MQ Administrator (MQADM)" on page 112.

5.2.3 The Application User (Otto, Eva, ...)

The application user needs access to queues, that his applications use. In the following example, the users Otto and Eva use the application AA which uses three queues.

```
CRTAUTL AUTL(AAUSER) TEXT('Users of Application AA')
GRTMQMAUT OBJ(TEST1) OBJTYPE(*Q) AUT(*USE) AUTL(AAUSER)
GRTMQMAUT OBJ(TEST2) OBJTYPE(*Q) AUT(*USE) AUTL(AAUSER)
GRTMQMAUT OBJ(TEST3) OBJTYPE(*Q) AUT(*USE) AUTL(AAUSER)

ADDAUTLE AUTL(AAUSER) USER(OTTO) AUT(*USE)
ADDAUTLE AUTL(AAUSER) USER(EVA) AUT(*USE)
```

There is also another way of doing this:

```
GRTMQMAUT OBJ(TEST1) OBJTYPE(*Q) USER(OTTO) AUT(*USE)
GRTMQMAUT OBJ(TEST2) OBJTYPE(*Q) USER(OTTO) AUT(*USE)
GRTMQMAUT OBJ(TEST3) OBJTYPE(*Q) USER(OTTO) AUT(*USE)
GRTMQMAUT OBJ(TEST1) OBJTYPE(*Q) USER(EVA) AUT(*USE)
GRTMQMAUT OBJ(TEST2) OBJTYPE(*Q) USER(EVA) AUT(*USE)
GRTMQMAUT OBJ(TEST3) OBJTYPE(*Q) USER(EVA) AUT(*USE)
```

You see that using an authorization list becomes more handy the more objects and users you have.

The application user also needs the library QMQM in the user library list. The APIs used in the MQ programming interface refer to objects in QMQM.

5.2.4 The Operator (QSYSOPR, MQOPR)

The standard AS/400 system operator, QSYSOPR, has authority to run the administration utility and the MQ service and channel commands.

This is how you set up QSYSOPR to run the admin utility:

```
CALL PGM(QMQM/AMQSADM4) PARM(QSYSOPR)
```

Only if you have revoked the use of QSYSOPR (by setting the password to *NONE), will you need another user profile for operator jobs (MQOPR). This user profile must be granted to do MQ operator tasks. Making a *SYSOPR user profile is not enough.

If you have saved the commands in authorization lists as we have shown in 5.2.1, “The MQ Administrator (MQADM)” on page 112, you only need to add the user (MQOPR) to the authorization list:

```
ADDAUTLE AUTL(MQSRV) USER(MQOPR) AUT(*USE)
ADDAUTLE AUTL(MQCHL) USER(MQOPR) AUT(*USE)

CALL PGM(QMQM/AMQSADM4) PARM(MQOPR)
```

Remember, before you back up or IPL an AS/400 you must end and then start the MQ manager and other MQ jobs. The operator doesn’t have to be authorized to all steps of closing down the MQM. The MQ manager has that authority. So if we want the operator to initiate such a task we can simply create a program that does this and allow the operator to run the task.

There are two ways to do that:

1. The program is compiled by MQADM with the parameter OWNER(*YES) and the operator is granted use of the program. This is called *adapted user profile*.

2. The second way we like better because it prevents misuse. Here the program is compiled as usual, but without OWNER(*YES). We create a JOBD (job description) that calls this program. The JOBD runs under the MQADM user profile. We authorize the operator to use the JOBD:

   ```
   CRTJOBD JOBD(ENDMQJD) JOBQ(QSYSNOMAX) USER(MQADM) RQSDTA('call endmq')
   GRTOBJAUT OBJ(ENDMQJD) OBJTYPE(*JOBD) AUT(*USE)
   ```

   When the operator submits a job (SBMJOB) with this JOBD he initiates the ending task of MQ and holds all the detailed authorities to MQM.

   ```
   SBMJOB JOBD(ENDMQJD)
   ```

   Access to the command SBMJOB is *PUBLIC but access to the JOBD that calls the ending program is granted to the operator.

Chapter 6. Running the Samples

Running the sample program is a good starting point to get acquainted with MQSeries for AS/400. You can do this at your AS/400 without connecting to other systems. To set up the MQSeries environment and to execute the samples do the following:

- Create the MQ Manager.
- Create the default environment.
- Create the sample objects.
- Compile the sample programs.
- Run the sample programs.

Samples are provided in the following languages: RPG, ILE-RPG, COBOL, ILE COBOL and ILE-C. In this chapter, we show how to work with the RPG samples.

The monitor programs AMQSERV4 and AMQSTRG4 are ready to use. We do not recompile them, since their source is not available in RPG.

The *MQSeries for AS/400 Application Programming Reference* manual describes the sample programs in detail. Our intention here is just to show you how to run them.

6.1 Creating the MQ Manager

This is done only once, since the AS/400 has only one instance of the MQ manager. If you decide later that you want a queue manager with another name you must delete this test queue manager first.

A detailed description on how to create the queue manager and its objects is in 4.5, “Creating the Queue Manager” on page 93.

To create a queue manager for the samples follow these steps.

1. Make sure that an MQM does not already exist!
2. Sign on as QSECOFR or as the MQ administrator created in Chapter 5, “Security Considerations” on page 107.
3. For this exercise we have chosen a name that we know you will not use: QM400.
4. Bring up the Create Message Queue Manager panel shown in Figure 64 on page 118 with the command:

© Copyright IBM Corp. 1998

CRTMQM

Do *not* press Enter; press F4 instead and then F11.

5. Type the name, the text, and the name of the message queue for undelivered messages. The choices for this example are shown in **bold.**

```
                    Create Message Queue Manager (CRTMQM)

 Type choices, press Enter.

 Message Queue Manager name . . . MQMNAME      > QM400_____________________________
_____________________
 Text 'description' . . . . . . . TEXT         > 'MQ manager for RALYAS4C'_____
_____________________________________
 Trigger interval . . . . . . . . TRGITV         999999999__
 Undelivered message queue  . . . UDLMSGQ        SYSTEM.DEAD.LETTER.QUEUE_______
_____________
 Default transmission queue . . . DFTTMQ         *NONE_______________________________
____________________
 Maximum handle limit . . . . . . MAXHDL         256________
 Maximum uncommitted messages . . MAXUMSG        10000______




                                                                          Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 64. Create Message Queue Manager

6. Press Enter to create the queue manager.
7. Create the default objects for the queue manager by calling the CL program that is supplied with MQSeries:

   CALL AMQSDEF4

If you made an error, delete the MQM now and create a new one. Below is an example:

```
DLTMQM ('thE.wroNg.NaMe')
CRTMQM F4 prompt
```

6.2 Creating the Environment for the Samples

The object definitions for the sample programs are in the member AMQSSAMP4 in the source file QMQMSAMP/QCLSRC. To create the definitions follow these steps:

1. Compile this sample program that defines the sample objects with the following command:

   ```
   CRTCLPGM PGM(QMQM/AMQSAMP4) SRCFILE(QMQMSAMP/QCLSRC)
   ```

2. Execute the program to create the objects:

   ```
   CALL QMQM/AMQSAMP4
   ```

3. Before you can do anything else to MQ you must start the MQ manager:

   ```
   STRMQM
   ```

After the MQM is started, you can view the new environment with the commands below. They will not work when MQM is not started.

```
DSPMQM
WRKMQMQ
WRKMQMPRC
WRKMQMOBJN
```

Figure 65 on page 120 is an example of the WRKMQMQ command.

Type 5 (display option) in front of the queue or process you want to look at. This example displays the panel in Figure 66 on page 120 which shows the queue attributes.

Note: You can also explore the commands by keying:

```
GO CMDMQM
```

This will show a menu with all MQ CL commands.

```
                              Work with MQM Queues

Type options, press Enter.
  2=Change   3=Copy    4=Delete   5=Display   6=Clear   14=Display authority
  15=Grant authority   16=Revoke authority

Opt  Name                                              Type  Text
5_   SYSTEM.DEFAULT.INITIATION.QUEUE                   *LCL  MQSeries Default
__   SYSTEM.DEFAULT.LOCAL.QUEUE                       *LCL
__   SYSTEM.DEFAULT.MODEL.QUEUE                       *MDL
__   SYSTEM.DEFAULT.REMOTE.QUEUE                      *RMT
__   SYSTEM.MQSC.REPLY.QUEUE                          *MDL  MQSeries reply qu
__   SYSTEM.SAMPLE.ALIAS                              *ALS  Sample alias queu
__   SYSTEM.SAMPLE.ECHO                               *LCL  queue for AMQSECH
__   SYSTEM.SAMPLE.INQ                                *LCL  queue for AMQSINQ
__   SYSTEM.SAMPLE.LOCAL                              *LCL  Sample local queu
__   SYSTEM.SAMPLE.REMOTE                             *RMT  Sample remote que
__   SYSTEM.SAMPLE.REPLY                              *LCL  General reply que
                                                                      More...
Parameters for options 2, 3, 5, 14, 15, 16 or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt      F5=Refresh   F6=Create   F9=Retrieve   F12=Cancel
F16=Repeat position to   F17=Position to          F20=Right     F21=Print
```

Figure 65. Work with MQM Queues

```
                               Display MQM Queue

Queue name . . . . . :   SYSTEM.DEFAULT.INITIATION.QUEUE

Queue type . . . . . :   *LCL
Text 'description' . . . . . :   MQSeries Default initiation queue

Put enabled  . . . . . :   *YES
Default message priority . . . :   0
Default message persistence  . :   *NO
Process name . . . . . :

Triggering enabled . . . . . :   *NO
Get enabled  . . . . . :   *YES
Sharing enabled  . . . . . :   *YES
Default share option . . . . . :   *YES
Message delivery sequence  . . :   *PTY
Harden backout count . . . . . :   *NO
                                                                      More...



F3=Exit   F12=Cancel   F21=Print
```

Figure 66. Display MQM Queue

6.3 Working with the RPG Sample Programs

Sign on as the programmer we created in 5.2.2, “The MQ Programmer (MQPGM)” on page 113.

If you use another user profile, remember that the user must have the authority to use (add and maybe delete) entries in the MQM (MQ manager) catalog (*CTLG) in order to create MQ queues and processes.

Check with the command DSPLIBL that the library QMQM *is in the user library list* and the library that shall contain your sample environment.

MQPGM *must* be the current library. Don′t use the QMQMSAMP or QGPL library as your current library. It is not forbidden, it is just a convention not to change IBM-supplied libraries.

6.3.1 Copy Source of the Sample Programs

Since we don′t want to change the samples library, we copy the RPG samples into the programmer′s library MQPGM before we compile or work with them.

1. Start the Program Development Manager with the command STRPDM.
2. In the subsequent panel, select 1 (work with libraries) and press Enter.
3. Type QMQM* in the Library field and press Enter.
4. Type 12 (option work with) in front of QMQMSAMP and press Enter.
5. In the next panel, scroll forward until you find QRPGSRC and type 3 (option copy) in front of the source file QRPGSRC as shown in Figure 67 on page 122.
6. Press Enter to copy the RPG source programs to your own program library MQPGM (Figure 68 on page 122).

```
                          Work with Objects Using PDM                 RALYAS4C

Library . . . . .   QMQMSAMP__       Position to . . . . . .   __________
                                      Position to type  . . . . .   __________

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display       7=Rename
  8=Display description        9=Save       10=Restore      11=Move ...

Opt  Object      Type        Attribute   Text
__   QLBLSRC     *FILE       PF-SRC      COBOL Samples
__   QMQSC       *FILE       PF-SRC      MQSC Samples
__   QRPGLESRC   *FILE       PF-SRC      ILE RPG/400 Samples
3_   QRPGSRC     *FILE       PF-SRC      RPG Samples
__   TEXT        *FILE       PF-SRC
__   QAM0150     *PRDLOD


                                                                       Bottom
Parameters or command
===> ____________________________________________________________________________
F3=Exit          F4=Prompt           F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry   F23=More options    F24=More keys
```

Figure 67. Work with Objects Using PDM

```
                                 Copy Objects

From library  . . . . . :   QMQMSAMP

Type the library name to receive the copied objects.

  To library  . . . . . .   MQPGM_____

To rename copied object, type New Name, press Enter.

Object      Type        New Name
QRPGSRC     *FILE       QRPGSRC___









                                                                       Bottom
F3=Exit     F5=Refresh     F12=Cancel     F19=Submit to batch
```

Figure 68. Copy Objects

6.3.2 Compile the RPG Programs

Press F3 to go back to the AS/400 Programming Development Manager (PDM) panel. From here you start the compile process:

1. Select 1 (work with libraries) and press Enter.
2. In the subsequent panel, Specify Libraries to Work With, enter MQPGM (your program library) as library and press Enter.
3. Type 12 in front of your library MQPGM and press Enter.
4. Scroll, and type 12 in front of the QRPGSRC source file and press Enter. This displays the panel shown in Figure 69 on page 124.
5. Start compiling the programs by typing 14 (option compile) in front of the programs, one after the other.
6. Press F4 and you will see the panel shown in Figure 70.

```
                      Create RPG/400 Program (CRTRPGPGM)

 Type choices, press Enter.

 Program  . . . . . . . . . . . . > AMQ1ECH4__    Name, *CTLSPEC
   Library  . . . . . . . . . . . >   MQPGM_____  Name, *CURLIB
 Source file  . . . . . . . . . . > QRPGSRC       Name, QRPGSRC
   Library  . . . . . . . . . . . >   MQPGM       Name, *LIBL, *CURLIB
 Source member  . . . . . . . . . > AMQ1ECH4      Name, *PGM
 Generation severity level  . . .   9___          0-99
 Text 'description' . . . . . . .   *SRCMBRTXT_____________________________________
_____

                           Additional Parameters

 Replace program  . . . . . . . . > *NO_          *YES, *NO




                                                                        Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display        F24=More keys
```

Figure 70. Create RPG/400 Program (CRTRPGPGM)

7. Specify the library for the object, here MQPGM, and press Enter.
8. After the compilation type WS (work with submitted jobs) in the option field (Opt).
9. Type DM in an option field to find out if the compile executed OK.

```
                          Work with Members Using PDM                RALYAS4C

File  . . . . . .   QRPGSRC___
  Library . . . .     MQPGM_____           Position to  . . . . .   __________

Type options, press Enter.
 2=Edit         3=Copy  4=Delete 5=Display       6=Print     7=Rename
 8=Display description  9=Save  13=Change text  14=Compile  15=Create module...

Opt  Member      Type        Text
14   AMQ1ECH4    RPG_______  Sample echo program_____________________________
__   AMQ1GBR4    RPG_______  Sample using Browse_____________________________
__   AMQ1GET4    RPG_______  Sample using MQGET______________________________
__   AMQ1INQ4    RPG_______  Sample using MQINQ______________________________
__   AMQ1PUT4    RPG_______  Sample using MQPUT______________________________
__   AMQ1REQ4    RPG_______  Sample using reply queue_________________________
__   AMQ1SET4    RPG_______  Sample using MQSET______________________________

                                                                       Bottom
Parameters or command
===> ____________________________________________________________________________
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

Figure 69. Work with Members Using PDM

If you get errors, check if the library list contains QMQM. The compiler copies from that library the RPG language source definitions such as the MQ message descriptor.

6.4 Executing the Sample Programs

Make sure that the MQ manager is started. It's best to go back to the first panel using F3. Then call the programs from the command line. Use this format:

```
CALL AMQ1xxx4
```

Then press the prompt key F4.

The reason for the prompt is that you need to enter the queue name into a 48-character long field. The prompt shows the 48-character field. Besides the queue name type a quote in the first and the last positions. The sample programs have been written to require this input.

The following examples show how to put messages on a queue, how to browse them and how to remove them.

6.4.1 Write Messages to a Queue

We use the sample program AMQ1PUT4 to write three messages to the queue SYSTEM.SAMPLE.LOCAL. This queue is one of the objects created in 6.2, “Creating the Environment for the Samples” on page 118.

To execute the sample program follow these steps:

1. On the command line, enter `CALL AMQ1PUT4` and press F4 (prompt). This displays the panel shown in Figure 71 on page 126.
2. Type `'SYSTEM.SAMPLE.LOCAL` and an ending quote (') in the last position of the first parameter.
3. Press Enter to run the program.
4. Next let us verify that there are messages on the queue. On the command line type the following command:

   ```
   WRKMQMQ QNAME(SYSTEM.SAMPLE.LOCAL)
   ```

   You don't have to key all 48 characters of the queue name. And you don't have to type them in uppercase either.
5. In the subsequent panel type 5 (option display) and press Enter.
6. In the next panel, Display MQM Queue, scroll to the bottom and look at the current queue depth. It should indicate that there are three messages in the queue.
7. You may look at the content of the queue, that is, view the messages in it with the following command:

   ```
   WRKMQMMSG QNAME(SYSTEM.SAMPLE.LOCAL)
   ```

   Figure 72 on page 126 lists the three messages.
8. The panel lets you display the *Message Description*, (header) and the *Message Data* by keying either 5 or 8 in front of the message. Figure 73 on page 127 shows an example of the message data.

```
                              Call Program (CALL)

 Type choices, press Enter.

 Program  . . . . . . . . . . . . > AMQ1PUT4__    Name
   Library  . . . . . . . . . . .     *LIBL_____  Name, *LIBL, *CURLIB
 Parameters . . . . . . . . . . . > 'SYSTEM.SAMPLE.LOCAL_____________________
     '
_____
               + for more values   ___________________________________________
______

                                                                          Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 71. CALL AMQ1PUT4

```
                         MQSeries Work with Messages

 Queue name . . . . . . .   SYSTEM.SAMPLE.LOCAL_________________________

 Type options, press Enter.
   4=Delete   5=Display Description   8=Display Data

 Opt   Date       Time       Type       UserId         Format     Size
  _    19980512   22070468   DATAGRAM   MQADM          MQSTR           60
  _    19980512   22070527   DATAGRAM   MQADM          MQSTR           60
  _    19980512   22070534   DATAGRAM   MQADM          MQSTR           60









                                                                          Bottom
 Command
 ===> ____________________________________________________________________________
 F3=Exit   F4=Prompt      F5=Refresh       F9=Retrieve   F12=Cancel
 F16=Repeat position to   F17=Position to   F21=Print
```

Figure 72. MQSeries Work with Messages

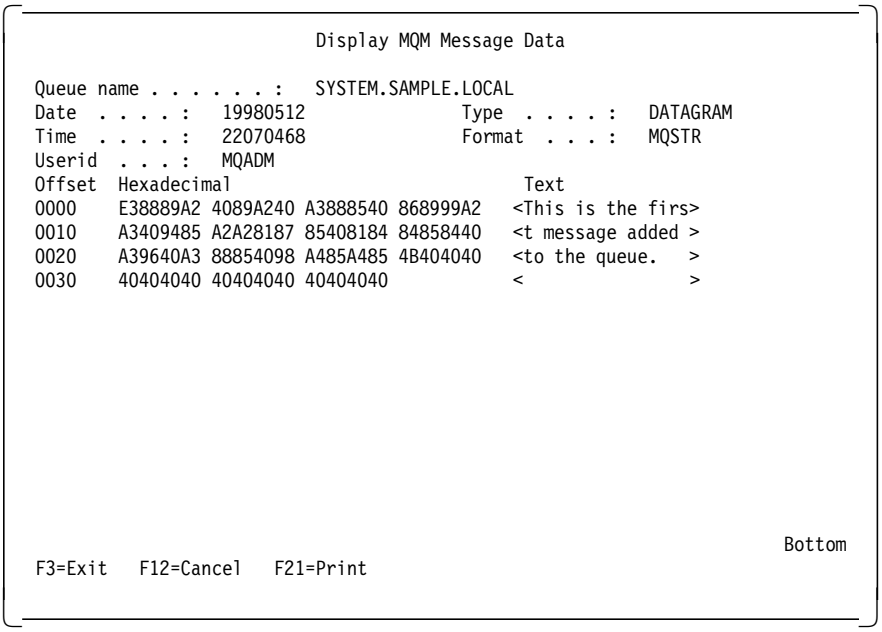


Figure 73. Display MQM Message Data

6.4.2 Browse Messages in a Queue

MQSeries provides the RPG sample program AMQ1GBR4 to browse messages in a queue. This program does not remove the messages.

Note: You cannot easily perform this function with standard AS/400 data queues.

To execute the sample program follow these steps:

1. On the command line, enter the following command:

   ```
   CALL PGM(AMQ1GBR4) PARM('SYSTEM.SAMPLE.LOCAL         ')
   ```

   Remember: Use prompt (F4), or retrieve a former call with F9.

   Type the correct program name and all 48 characters of the queue name.

2. In order to look at the result of the program execution, type WRKSPLF and 5 in front of the spool file AMQ1GBR4.

Also look again at the queue SYSTEM.SAMPLE.LOCAL, and the messages.

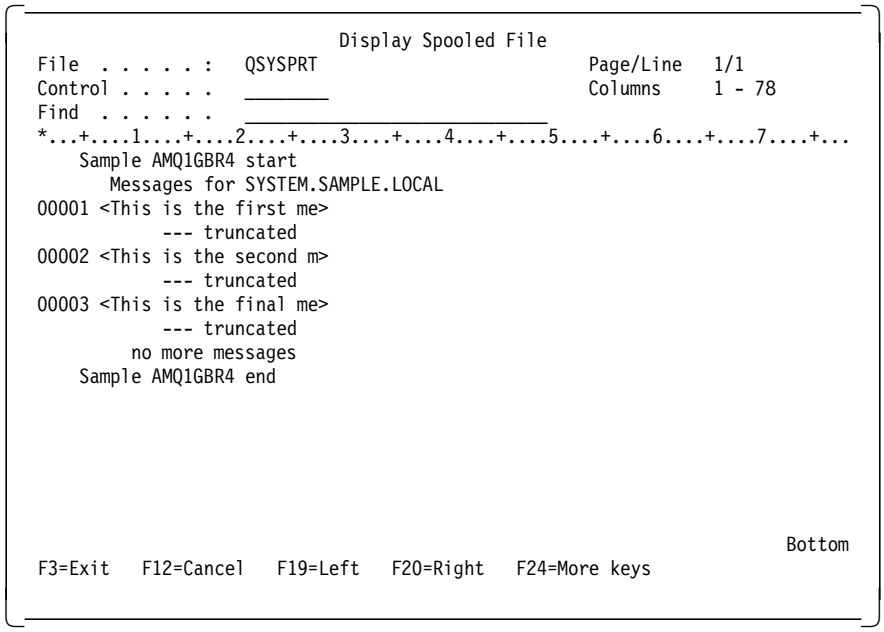


Figure 74. Display Spooled File

6.4.3 Get Messages from a Queue

The sample program AMQ1GET4 reads and removes messages from a queue. The program does what is referred to as a destructive get. To execute the program follow these steps:

1. On the command line, enter:

   ```
   CALL PGM(AMQ1GET4) PARM('SYSTEM.SAMPLE.LOCAL         ')
   ```

   Remember to use prompt (F4), or retrieve a former call with F9, and key the right program name, all for the purpose of getting the 48-character long parameter field for the sample queue.

2. Once again look at the queue with WRKMQMQ SYSTEM.SAMPLE.LOCAL, and see that the messages have disappeared. Use the command WRKSPLF and type 5 in front of AMQ1GET4. This displays the messages shown in Figure 75 on page 129.

```
                          Display Spooled File
File  . . . . . :   QSYSPRT                      Page/Line   1/1
Control . . . . .  ________                      Columns     1 - 78
Find  . . . . . .  ____________________
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
    Sample AMQ1GET4 start
 message <This is the first message added to the queue.              >
 message <This is the second message.                                >
 message <This is the final message.                                 >
         no more messages
    Sample AMQ1GET4 end
                                                                    Bottom
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

Figure 75. Display Spooled File

6.5 Triggering a Program

In this section, we explain how to trigger an application program when a message arrives in a queue.

6.5.1 Create a Sample Program

Write the CL program DTA2 in A.4, “DTA2 - Put Date and Time in a Data Area” on page 188 and compile and test it following these steps:

- CRTSRCPF FILE(SRC) creates the file SRC in library MQPGM.
- STRPDM
- Type 1, work with libraries.
- Enter MQPGM as library.
- Type 12 (work with) for library MQPGM.
- Choose 12 at Object SRC, Type *FILE, Attribute PF-SRC.
- Select F6 (create).
- Select F6 SRCMBR(DTA2) TYPE(CLP).
- Type he program, remember prompt, finish with F3.

- Type 14 to compile to library MQPGM.
- Test the program with C and F4 (call).
- Key a parameter 'xxxx'.
- DSPDTAARA DTAARA(DTA2), watch the date and time.

6.5.2 Create a Triggered Queue

Create the test queue TEST2:

```
CRTMQMQ QNAME(TEST2) QTYPE(*LCL) TEXT('Triggered queue')
        PRCNAME(PROC2) TRGENBL(*YES) INITQNAME(SYSTEM.SAMPLE.TRIGGER)
```

Naturally, you will use prompts to do this. Watch the quotes in the prompt. If you keep the quotes, type in uppercase.

When this queue is triggered (by a monitoring program) the *Process* PROC2, will be started.

6.5.3 Create a Process

The process describes what the program is be called (and how). Create this process:

```
CRTMQMPRC PRCNAME(PROC3) TEXT('calls MQPGM/DTA2')
          APPID('MQPGM/DTA2')
```

When you use prompts, keep the quotes where they are. APPID must have quotes around the program to call if it is qualified. If you just write DTA2, you don't need the quotes. In this example we don't use *User data* or *Environment data*.

6.5.4 Start Program that Monitors the Initiation Queue

Remember the 48 long parameter, when calling the program that monitors the triggering:

```
CALL PGM(AMQSERV4) PARM('SYSTEM.SAMPLE.TRIGGER             ')
```

This should normally be submitted to batch. But in order to see it in action we execute the program in a 5250 session (it will not end, if not cancelled).

This is the program that will actually CALL MQPGM/DTA2.

6.5.5 Put Messages into Triggered Queue

Sign on at a new 5250 session with the same user as in the other 5250 session.

First clear the message queue TEST2 and then type WRKMQMMSG TEST2 and clear messages, if there are any.

Use the put program to put 3 messages in the queue:

```
CALL PGM(AMQ1PUT4) PARM('TEST2                                    ')
```

6.5.6 Watch the Monitor Program

The panel should show the following:

```
   ...>
   QMQM/AMQSERV4
   ...>
   QMQM/AMQSERV4
   ...>
   CALL PGM(MQPGM/DTA2) PARM('TM     1TEST2
         PROC2
                                              MQPGM/DTA2




                    ');
   ...>


 ===> ____________________________________________________________________
 F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
 F18=Bottom  F19=Left   F20=Right F21=User Window
```

Figure 76. AMQSERV4 Writing to STDOUT

The log shows that the program calls MQPGM/DTA2. You can cancel the program by pressing System Request and typing 2. If you cannot find System Request, then stop the program by disabling the get from the queue.

Change the queue from the other 5250 session:

```
CHGMQMQ QNAME(SYSTEM.SAMPLE.TRIGGER) GETENBL(*NO)
```

Then enable the get again when AMQSERV4 has stopped executing.

```
CHGMQMQ QNAME(SYSTEM.SAMPLE.TRIGGER) GETENBL(*YES)
```

Display the time stamp in dataarea DTA2 (done by the program MQPGM/DTA2):

```
DSPDTAARA DTAARA(DTA2)
```

Note: If you don't start the monitoring program, the initiation message will stay in the initiation queue. Try it, and try to display the queue with the command:

```
WRKMQMMSG QNAME(SYSTEM.SAMPLE.TRIGGER)
```

No message is displayed. It is obviously not displayed by the command. But there is a message in the queue. If you start the monitor the action of the process will take place.

6.5.7 Monitoring in Batch

You can test the monitoring program without occupying a 5250 session:

```
SBMJOB CMD(CALL PGM(AMQSERV4) PARM('SYSTEM.SAMPLE.TRIGGER    ')) +
               JOBQ(QSYSNOMAX)
```

Watch it executing:

```
WRKACTJOB SBS(QSYSWORK)
```

```
                            Work with Active Jobs                    RALYAS4C
                                                          05/13/98  01:07:13
CPU %:    1.5     Elapsed time:   00:03:11     Active jobs:   130

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...

Opt  Subsystem/Job  User        Type  CPU %  Function        Status
__   QSYSWRK        QSYS        SBS      .0                  DEQW
__     AMQALMP4     QMQM        BCH      .0  PGM-AMQALMP4    DEQW
__     AMQMCPRA     QMQM        BCH      .0  PGM-AMQMCPRA    DEQW
__     FSIOP        QSYS        BCH      .0  PGM-QFPAMONB    TIMW
__     MQPGM        MQADM       BCH      .1  PGM-AMQSERV4    DEQW
__     QAPPCIPX     QSYS        BCH      .0                  TIMW
__     QAPPCTCP     QSYS        BCH      .0  PGM-QZPAIJOB    TIMW
__     QCQEPMON     QSVMSS      BCH      .0  PGM-QCQEPMON    MSGW
__     QCQRCVDS     QSVMSS      BCH      .0  PGM-QCQAPDRM    MSGW
                                                                More...
Parameters or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt        F5=Refresh   F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F14=Include   F24=More keys
```

Figure 77. Work with Active Jobs, PGM-AMQSERV4

Do not submit AMQSERV4 to QBATCH. The job queue QBATCH is often specified, so it will only take one job at a time. If you submit a never ending program like this, then new jobs will be trapped in line of the job queue waiting for AMQSERV4 to end. Submit to job queue QSYSNOMAX that serves the subsystem QSYSWRK.

There is also an alternative program to AMQSERV4. The program AMQSTRG4 does the same job, but by submitting the process program to batch, instead of calling it. The submit will allow parallel jobs (which AMQSERV4 does not), however, a submit has an additional overhead. AMQSTRG4 is hardcoded to submit the program to the QBATCH job queue, thus executing in the QBATCH subsystem.

Use the following commands:

```
SBMJOB CMD(CALL PGM(AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER     ')) +
               JOBQ(QSYSNOMAX)
CLRMQMQ QNAME(TEST2)
CALL PGM(AMQ1PUT4) PARM('TEST2                                ')
```

Watch the submit when messages are put to TEST2 (after clearing):

```
WRKACTJOB
```

```
                            Work with Active Jobs                    RALYAS4C
                                                              05/13/98  01:19:50
CPU %:    1.7     Elapsed time:   00:15:48     Active jobs:   131

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...

Opt  Subsystem/Job  User        Type  CPU %  Function        Status
__   QBATCH         QSYS        SBS      .0                 DEQW
__     MQPGM        MQADM       BCH      .0  PGM-DTA2       RUN
__   QCMN           QSYS        SBS      .0                 DEQW
__     RAL5494A     QUSER       EVK      .0                 EVTW
__   QCTL           QSYS        SBS      .0                 DEQW
__     QSYSSCD      QPGMR       BCH      .0  PGM-QEZSCNEP   EVTW
__   QINTER         QSYS        SBS      .0                 DEQW
__     DSP06        WILLEMS     INT      .0  CMD-TELNET     DEQW
__     DSP07        WILLEMS     INT      .0  CMD-WRKCFGSTS  DSPW
                                                                More...
Parameters or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt        F5=Refresh   F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F14=Include   F24=More keys
```

Figure 78. Work with Active Jobs, PGM-DTA2

This technique takes more time, because a new job must be initiated. However, the next job doesn't have to wait for the first one to finish.

If you call the program that puts a message in the triggered queue, but without running the monitoring programs (AMQSERV4 or AMQSTRG4) the triggering message is left in the queue SYSTEM.SAMPLE.TRIGGER. If you try to display the message with DSPMQMMSG, you cannot see it.

6.6 More Sample Programs

The sample programs AMQ1REQ, AMQ1SET, AMQ1INQ, and AMQ1ECHO are described in the *MQSeries for AS/400 Application Programming Reference*. The application logic is as follows:

1. The request program AMQ1REQ puts messages in the queue QA and waits for messages to arrive in queue QB.
2. After the messages have arrived in the triggered queue QA the program PA is triggered to process it.
3. The triggered program returns replies in queue QB.

If the first queue (QA) is SYSTEM.SAMPLE.INQ then the program triggered should be AMQ1INQ4. SYSTEM.SAMPLE.ECHO triggers the program AMQ1ECH4, and SYSTEM.SAMPLE.SET triggers AMQ2SET4.

The samples are set up to start the (C programs) AMQSINQA, AMQSECHA, and AMQSSETA. When we want to execute the RPG programs compiled earlier in this chapter, we have to change the processes to point to the programs AMQ1INQ4, AMQ1ECH4, and AMQ1SET4 instead. You do this with the command:

```
CHGMQMPRC
```

That displays the panel shown in Figure 79 on page 135.

The processes to change are:

- SYSTEM.SAMPLE.ECHOPROCESS
- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS

```
                        Change MQM Process (CHGMQMPRC)

 Type choices, press Enter.

 Process name . . . . . > 'SYSTEM.SAMPLE.ECHOPROCESS'
 Text 'description' . . . . . 'trigger process for AMQSECHA
 Application type . . . . . *OS400      65536-999999999, *OS400...
 Application identifier . . . . . 'AMQ1ECH4
 User data  . . . . . '
                                                                    More...
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 79. Change MQM Process (CHGMQMPRC)

You can test the programs with the commands below. The result will be in the spool file.

```
SBMJOB CMD(CALL PGM(AMQSERV4) PARM('SYSTEM.SAMPLE.TRIGGER
          ')) JOBQ(QSYSNOMAX)
CALL PGM(AMQ1REQ4) PARM('SYSTEM.SAMPLE.ECHO               ')
CALL PGM(AMQ1REQ4) PARM('SYSTEM.SAMPLE.INQ                ')
CALL PGM(AMQ1REQ4) PARM('SYSTEM.SAMPLE.SET                ')
CHGMQMQ QNAME(SYSTEM.SAMPLE.TRIGGER) GETENBL(*NO)
CHGMQMQ QNAME(SYSTEM.SAMPLE.TRIGGER) GETENBL(*YES)
WRKSPLF
```

If you use a separate 5250 session for AMQSERV4 instead of submitting, you can follow the process on the screen.

If you experience problems, look for the following in the spool file:

```
Sample AMQ1PUT4 start
Target queue is TEST2                            st2"       c
MQOPEN ended with reason code   000002085
 Unable to open queue for output
Sample AMQ1PUT4 end
```

Reason code 2085 (see page 291 in the *Application Programmers Reference*) tells you that the problem is an unknown object name. Here the queue has not been entered as a 48-character string. So the queue name (TEST2) has the characters (st2″ c) added to it.

6.7 Inspecting Some Code

Use STRPDM to inspect the sample programs, and look also at the compilation lists.

Notice the /COPY statement that copies the source of the message data structures into the program.

You will find this piece of code, before the GET in the AMQ1GET program:

```
C           CCODE     DOWNECCFAIL
 * option is to wait up to 15 seconds for next message
C                     Z-ADDGMWT      GMOPT            wait
C                     ADD  GMCONV    GMOPT            convert
C                     Z-ADD15000     GMWI             up to 15sec
 *
 **  MsgId and CorrelId are selectors that must be cleared
 **  to get messages in sequence, and they are set each MQGET
C                     MOVELMINONE    MDMID
C                     MOVELCINONE    MDCID
 **  clear buffer because MQGET only fills to length of message
C                     MOVEL*BLANKS   BUFFER
 *  call ...
C                     Z-ADDMQGET     CID
C                     CALL 'QMQM'
C                     PARM           CID     90
C                     PARM           HCONN   90
C                     PARM           HOBJ    90
C                     PARM           MQMD
C                     PARM           MQGMO
C                     PARM           BUFLEN  90
C                     PARM           BUFFER 60
C                     PARM           MESLEN  90
C                     PARM           CCODE   90
C                     PARM           REASON  90
```

Adding the constant CMCONV to the option field CMOPT enables ASCII to EBCDIC translations which are necessary if the messsages come from a Windows NT machine. If you look at AMQ1GBR4 you will see that the code is missing; therefore, AMQ1GBR4 will not translate.

You may consider adding the *well-behaved-program-option* to CMOPT. To behave well means that the program will not hang forever in the GET if MQ manager is taken down (quiescing). This is the get message option fail_if_quiescing, GMFIQ.

```
C                   Z-ADDGMWT      GMOPT           wait
C                   ADD  GMCONV    GMOPT           convert
C                   ADD  GMFIQ     GMOPT           convert
```

6.8 Syncpoint and AS/400 Commit Control

You should consider your data to be as safe in an MQ queue as you know it is in an AS/400 database file. You should syncronize your transactions, for example, if you read records from a file and put them in a queue, then data from a single transcaction should be:

- In the queue and not in the file, if the transcaction is completed.
- In the file and not in the queue, if the transcaction has failed.

The programs AMQ1PUT4 and AMQ1GET4 do not put or get under syncpoint.

Make the following changes to provide transaction syncronization:

1. Copy the source program for AMQ1GET4 and give it the name GETEX.
2. Find the place in the program where GET-module-options are entered:

```
⋮
     * option is to wait up to 15 seconds for next message
    C                   Z-ADDGMWT      GMOPT           wait
    C                   ADD  GMCONV    GMOPT           convert
    C                   Z-ADD15000     GMWI            up to 15sec
⋮
     *  call ...
    C                   Z-ADDMQGET     CID
    C                   CALL 'QMQM'
    C                   PARM           CID     90
    C                   PARM           HCONN   90
    C                   PARM           HOBJ    90
    C                   PARM           MQMD
⋮
```

3. Add the instruction:

```
    C                   ADD  GMSYP     GMOPT           GET U SYNCP
```

4. The GETEX program now looks like this:

```
     * option is to wait up to 15 seconds for next message
    C                   Z-ADDGMWT      GMOPT            wait
    C                   ADD  GMCONV    GMOPT            convert
    C                   ADD  GMSYP     GMOPT            GET U SYNCP
    C                   Z-ADD15000     GMWI             up to 15sec
```

To get comfortable with the sequence of operations, test the following scenario:

```
CRTMQMQ QNAME(TESTY) QTYPE(*LCL) DFTMSGPST(*YES)
                              /* Create a local queue              */

CLRMQMQ QNAME(TEST2)          /* Clear the TEST2 queue             */

CALL PGM(AMQ1GET4) PARM('TEST2                    ')
                              /* Put 3 messages in the queue TEST2 */

WRKMQMMSG QNAME(TEST2)        /* Display the messages              */

STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)  /* Note:  NOT default options */

CALL PGM(GETEX) PARM('TEST2                   ')
                              /* Gets all messages under syncpoint */

WRKMQMMSG QNAME(TEST2)        /* No messages                       */

ROLLBACK

WRKMQMMSG QNAME(TEST2)        /* The messages are back again       */

CALL PGM(AMQ1GET4) PARM('TEST2                    ')
                              /* Put 3 messages in the queue TEST2 */

WRKMQMMSG QNAME(TEST2)        /* Display the messages              */

CALL PGM(GETEX) PARM('TEST2                   ')
                              /* Gets all messages under syncpoint */

WRKMQMMSG QNAME(TEST2)        /* No messages                       */

COMMIT

WRKMQMMSG QNAME(TEST2)        /* The queue  is empty               */

DSCMQM                        /* Must be done before ENDCMTCTL     */
```

```
ENDCMTCTL                       /* Ends commit control              */
```

These operations can be tested together with the commit/rollback example from 3.14.3, “Securing Data between Transactions” on page 80. The program PGMA and the file FILA are in Appendix A, “Sample Programs” on page 187. The example above may now be carried out this way:

```
CLRMQMQ QNAME(TEST2)            /* clear the TEST2 queue            */

CALL PGM(AMQ1GET4) PARM('TEST2                    ')
                                /* put 3 messages in the queue TEST2 */

WRKMQMMSG QNAME(TEST2)          /* display the messages             */

STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
                                /* note:  NOT default options       */

CALL PGM(GETEX) PARM('TEST2                  ')
                                /* Gets all messages under syncpoint */

WRKMQMMSG QNAME(TEST2)          /* no messages                      */

CALL PGM(MYAPP/PGMA)            /* adds 3 records to file FILA      */

DSPPFM FILE(MYAPP/FILA)         /* see it yourself                  */

ROLLBACK

WRKMQMMSG QNAME(TEST2)          /* the messages are back again      */

DSPPFM FILE(MYAPP/FILA)         /* the added records are gone       */

CALL PGM(AMQ1GET4) PARM('TEST2                    ')
                                /* put 3 messages in the queue TEST2 */

WRKMQMMSG QNAME(TEST2)          /* display the messages             */

CALL PGM(GETEX) PARM('TEST2                  ')
                                /* Gets all messages under syncpoint */

WRKMQMMSG QNAME(TEST2)          /* no messages                      */

CALL PGM(MYAPP/PGMA)            /* adds 3 records to file FILA      */

DSPPFM FILE(MYAPP/FILA)         /* see it yourself                  */
```

```
COMMIT

WRKMQMMSG QNAME(TEST2)          /* the queue is empty               */

DSPPFM FILE(MYAPP/FILA)         /* data is in the file              */

DSCMQM                          /* must be done before ENDCMTCTL    */

ENDCMTCTL                       /* ends commit control              */
```

Note: The standard AS/400 operations for commit control also work for MQSeries for AS/400. Therefore, you don't need the MQ operations MQBEGIN, MQCMIT, and MQBACK.

You may change the AMQ1PUT4 program to put messages under syncpoint. The statement to add is:

```
     C                   ADD  PMSYP     PMOPT            PUT U SYNCP
```

This discussion shows that the information is either safe in an MQ queue or in a database file in the same AS/400.

Chapter 7. AS/400 Communicating with Other Systems

The reason for you to install MQSeries for AS/400 is probably because you want to exchange data in a network with a non-AS/400 system.

In this book, we discuss very briefly the communications setup. There are good redbooks available that explain this topic in detail. *AS/400 Communications Examples III*, GG24-4386 is a good example.

The communcation setup for MQSeries for AS/400 is described, in detail, in the manual *MQSeries Intercommunications*, SC33-1872-01. You will have to consult communications specialists for the platforms the AS/400 will communicate with and agree on communications information, such as node names, network names and much more.

The following example explains the MQ definitions required to connect a queue manager in an AS/400 with a queue manager in a Windows NT system using TCP/IP.

7.1 AS/400 Communications Objects

In order to find the actual hardware installed execute the command:

```
WRKHDWRSC TYPE(*CMN)
```

This displays the panel shown in Figure 80 on page 142

Communication objects are:

Line description
: Refers to a communication resource, an adapter card with a communication cable. A token-ring is an example. It contains information close to the LINE macro in VTAM.

Control description
: Refers to the line description it is connected to. It has information similar to the PU macro in VTAM. It describes the remote site.

Device description
: Refers to the control description of the remote unit (CPU or controller) that it is part of. This information is similar to the LU macro of VTAM.

Mode description
: Refers to the device/LU logical unit that runs the session. It has also an equivalent in VTAM.

© Copyright IBM Corp. 1998

```
                    Work with Communication Resources
                                                          System:   RALYAS4C
Type options, press Enter.
  5=Work with configuration descriptions   7=Display resource detail

Opt  Resource        Type  Status               Text
 _   CMB01           9162  Operational          Combined function IOP
 _     LIN01         2612  Operational          Comm Adapter
 _       CMN01       2612  Operational          V.24 Port Enhanced
 _   CCO2            2617  Not detected         Combined function IOP
 _     LIN03         2617  Not detected         LAN Adapter
 _       CMN03       2617  Not detected         Ethernet Port
 _     LIN09         605A  Not detected         Virtual Controller
 _   CCO3            2623  Operational          Comm Processor
 _     LIN04         2609  Operational          Comm Adapter
 _       CMN04       2609  Operational          V.24 Port
 _       CMN05       2609  Operational          V.24 Port
 _     LIN05         2613  Operational          Comm Adapter
 _       CMN06       2613  Operational          V.35 Port
 _     LIN06         2614  Operational          Comm Adapter
 _       CMN07       2614  Operational          X.21 Port
                                                                      More...
F3=Exit   F5=Refresh   F6=Print   F12=Cancel
```

Figure 80. Work with Communication Resources

For LU 6.2 communication in the AS/400, the device description and the controller desciption are often automatically defined. Although you don't see it these objects are in the library QSYS.

SNA information about your AS/400 system as an SNA node is in the *Net Attribute* for your system. Net attributes can be displayed by the command:

```
DSPNETA
```

and changed with:

```
CHGNETA
```

Some of the importent network attributes are:

- Current system name
- Local network ID
- Local control point name
- Default local location
- Default mode
- APPN node type

TCP/IP communication configuration is reached through the menu:

```
GO TCPADM
```

7.2 MQSeries for AS/400 Using SNA

If you already have RJE, SNADS, 3270 applications or OPC tracker running, you have the SNA definitions in your AS/400 working against an IBM host (MVS, CICS).

If you have ClientAccess/400 running LU 6.2 SNA to a number of PCs then the definitions already exist. You may already have Communication Manager in place for OS/2.

What MQSeries for AS/400 adds to this is a *Routing Entry* to a communications subsystem of AS/400: QCMN or QSNADS (but not both). This routing entry starts the program that serves the MQ channels with data from the SNA communication. The CL program AMQCRC6A starts the program AMQCRS6A.

Here is an example:

Enter ADDRTGE, press F4 and then F11. This displays the panel in Figure 81.

```
                          Add Routing Entry (ADDRTGE)

 Type choices, press Enter.

 Subsystem description  . . . . . SBSD           QSNADS
   Library  . . . . . . . . . . .                  *LIBL_____
 Routing entry sequence number  . SEQNBR         400___
 Comparison data:                 CMPVAL
   Compare value  . . . . . . . .                CKRC_____________________________
_____________________________________
   Starting position  . . . . . .                1_____
 Program to call  . . . . . . . . PGM            AMQCRC6A__
   Library  . . . . . . . . . . .                  QMQM______
 Class  . . . . . . . . . . . . . CLS            *SBSD_____
   Library  . . . . . . . . . . .                  __________
 Maximum active routing steps . . MAXACT         *NOMAX
 Storage pool identifier  . . . . POOLID         1_____




                                                                          Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 81. Add Routing Entry

SNA communication programs in AS/400 have two different styles:

ICF Intercommunications Facility, AS/400 only, and often used in RPG programs

CPI-C Common Programming Interface - Communications

The SNA communications interface is the same for all SNA communication platforms. A central point in CPI-C is the *CPI-C Side Information*.

The *compare value* of the routing entry must match the *transaction program* of the side information in the AS/400 and at the remote site.

The *remote location* must match the remote location in the device description (if you create it) or in the configuration list mentioned below (if the device description is automatically created).

This is an example of creating the CSI (CPI-C side information) for AS/400:

```
                    Create Comm Side Information (CRTCSI)

Type choices, press Enter.

Side information . . . . . CSI          > XX_____
  Library  . . . . . .                  >   QGPL_____
Remote location  . . . . . RMTLOCNAME   > HOSTXX__
Transaction program  . . . . . TNSPGM   > 'CKRC_____________________________
_____________________________________________'

Text 'description' . . . . . TEXT       > *BLANK_____________________________
_____________________

                         Additional Parameters

Device . . . . . DEV          > *LOC______
Local location . . . . . LCLLOCNAME   > *LOC____
Mode . . . . . MODE         > #INTER__
Remote network identifier  . . . RMTNETID     > *LOC____
Authority  . . . . . AUT            *LIBCRTAUT
                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 82. Create Comm Side Information

- The user QMQM must be known to RACF (for MVS communication).
- *Side information name* must match *Connection name* in the Sender Channel.
- Transaction program name and mode name must match the names at the remote site.

Here is a definition example for CM/2:

```
DEFINE_TP  TP_NAME(AMQCRS6A)
PARM_STRING(-n AMQCRS6A)
PIP_ALLOWED(NO)
FILESPEC(c: mqm bin amqcrs6a.exe)
CONVERSATION_TYPE(ANY_TYPE)
CONV_SECURITY_RQD(NO)
SYNC_LEVEL(EITHER)
TP_OPERATION(NONQUEUED_AM_STARTED)
PROGRAM_TYPE(BACKGROUND)
RECEIVE_ALLOCATE_TIMEOUT(INFINITE);
```

You may either define controller and device descriptions at the AS/400, or you may add a communication list entry to the remote location list and have the above definitions defined automatically by the first interaction with the remote site (vary communication on).

If you add a communication list entry use this command:

```
ADDCFGLE TYPE(*APPNRMT) APPNRMTE((HOSTXX))
```

Prompt with F4 to be sure to provide the correct values for remote location name and network, local location name, remote control point, etc.

7.3 MQSeries for AS/400 Using TCP/IP

Again, you probably have TCP/IP running, and you may not get much help from communications manuals, or communications people.

TCP/IP is a lot easier to handle than SNA.

Among other things you must specify:

- The IP address and subnet mask of your communication line
- The loopback interface
- A default route
- Other routing information
- Your AS/400 name and domain in the host information or in a name server
- Remote site's corresponding IP addresses and host name (with domain) in the host table or in a name server (inside AS/400 or in the network)

You may verify the setup at the menu reached with `GO TCPADM`.

When you have the IP address (or a name, known by a name server) you can use the PING operation to verify the connection.

A program called the *listener* must be started in order to get messages from the communication line to the receiving MQ channel. The AS/400 command is STRMQMLSR and the listener program is AMQCLMAA executing in the subsystem QSYSWRK.

The listener is a *Host service* like telnet. Telnet and many other TCP/IP services are started by the Start TCP/IP Server command (STRTCPSVR), or they may be started with *Start TCP/IP* (STRTCP), if the service is configured for autostart. The listener cannot be started this way. You have to use the STRMQMLSR command.

The IP port 1414 has been assigned to the listener. You may change this. The change is done in the member QMINI in the source file QMINI in the library MQMQDATA. We show you an example with port 1414 or the default. Using the default port 1414 the file member may just as well be empty.

```
TCP:
    PORT=1414
```

7.4 Example: Connecting MQSeries for AS/400 to Windows NT

The purpose of this simple example is to make you feel comfortable connecting queue managers running in two different systems, such as an AS/400 and Windows NT.

MQSeries uses name resolution. It is essential that MQ object names are specified correctly in both systems. The names are case sensitive.

Table 9 shows what definitions are necessary in both systems.

Table 9. Definitions for Connecting Windows NT to AS/400

Objects	**AS/400**	**Windows NT**
MQM name	QM400	NT1
Queues	NT1 (local tmq) N1 (remote) J1 (local)	QM400 (local tmq) N1 (local) J1 (remote)
Channels	QM400.NT1 (sender) NT1.QM400 (recv)	QM400.NT1 (receiver) NT1.QM400 (sender)

7.4.1 Defining the AS/400 MQ Environment

There are two ways to apply the definitions to a queue manager in an AS/400:

- With CL commands
- Using MQSC commands

7.4.1.1 Defining the Environment with CL Commands

We use the CL commands of MQSeries for AS/400. In this example, we use them interactively. However, you may gather them in a CL program, if you wish. In that case, you would probably specify REPLACE(*YES), so you can run it several times.

```
CRTMQM  QM400                    /* (if it was not done before)
CALL QMQM/AMQSDEF4               /* (standard definitions)
CALL QMQM/AMQSADM4 PARM(MQADM)   /* (making MQADM user of STRMQMADM)
                                 /*
CRTMQMQ QNAME(NT1) QTYPE(*LCL) +
        USAGE(*TMQ)
CRTMQMQ QNAME(N1) QTYPE(*RMT)
CRTMQMQ QNAME(J1) QTYPE(*LCL)

CRTMQMCHL CHLNAME(QM400.NT1)       +
          CHLTYPE(*SDR)            +
          CONNAME('9.24.104.116')  +
          TQNAME('NT1                                          ')
CRTMQMCHL CHLNAME(NT1.QM400) +   /* (TCP/IP is default)
          CHLTYPE(*RCVR)
```

7.4.1.2 Defining the Environment Using MQSC

Alternatively, you can use MQSC commands to define the MQ objects. Type them into the new member NTEX2 in a source file MQLIB/TEXT.

In the example shown in Figure 83 on page 148, we use the short synonyms for the MQSC commands. An example with more readable commands is in Chapter 2, “MQSeries Overview” on page 7. The commands are described, in detail, in the *MQSeries Command Reference*, SC33-1369.

First verify the syntax:

```
STRMQMMQSC SRCMBR(NTEX2) SRCFILE(MQLIB/TEXT) OPTION(*VERIFY)
```

Look at the error log with WRKSPLF. Display (5) the last spool file, which is probably in QSYSPRT printer/output (if not already out on paper).

```
****definitions for use on AS/400 side using STRMQMMQSC

def chl (NT1.QM400) chltype(rcvr) trptype(tcp)
******receiver channel that matches sender on NT***

def chl(QM400.NT1) chltype(sdr) trptype(tcp) conname(9.24.104.116) +
      xmitq(NT1)
******sender channel definition that matches rcvr on NT***

def ql(NT1) usage(xmitq)
******definition of transmission queue for data going to NT***

def qr(N1) rname(N1) rqmname(NT1)
******remote queue definition pointing to queue on NT***

def ql(J1)
******define local target queue on AS/400***
```

Figure 83. STRMQMMQSC Definitions

Figure 84 shows part of an error report.

```
       : ****definitions for use on AS/400 side using STRMQMMQSC
       :
     6 : def chl (NT1.QM400) chltype(rcvr) trytype(tcp)
AMQ8405 Syntax error detected at or near end of command segment below:-
def chl (NT1.QM400) chltype(rcvr) t
AMQ8427 Valid syntax for the MQSC command:
  DEFINE CHANNEL(channel-name) CHLTYPE( RCVR )
     [ BATCHSZ(integer) ]                [ DESCR(string) ]
     [ HBINT(integer) ]                  [ LIKE(channel-name) ]
     [ MAXMSGL(integer) ]                [ MCAUSER(string) ]
     [ MRDATA(string) ]                  [ MREXIT(string)]
     [ MRRTY(integer) ]                  [ MRTMR(integer) ]
     [ MSGDATA(string) ]                 [ MSGEXIT(string) ]
     [ NOREPLACE | REPLACE ]             [ NPMSPEED( NORMAL | FAST ) ]
     [ PUTAUT( DEF | CTX ) ]             [ RCVDATA(string) ]
     [ RCVEXIT(string) ]                 [ SCYDATA(string) ]
     [ SCYEXIT(string) ]                 [ SENDDATA(string) ]
     [ SENDEXIT(string) ]                [ SEQWRAP(integer) ]
     [ TRPTYPE( LU62 | TCP ) ]
```

Figure 84. STRMQMMQSC Error Report

Correct the error, TRPTYPE not TRYTYPE, verify again and run:

```
STRMQMMQSC SRCMBR(NTEX2) SRCFILE(MQLIB/TEXT) OPTION(*RUN)
```

7.4.2 Defining the Windows NT MQ Environment

You must create a Windows NT user, for example, MQADM. MQADM can perform MQ administration jobs on the Windows NT machine via the AS/400.

In the Windows NT machine, create the queue manager NT1 with this command:

```
crtmqm -q  NT1
```

Notes:

1. -q makes NT1 the default queue manager.
2. The command creates the queue manager and the default objects.

Next, start the queue manager with the command strmqm.

Note: If NT1 is not the default MQ manager you have to specify its name when you start it:

```
strmqm NT1
```

The we have to define the objects that correspond with the definitions in the AS/400. You can create the queues and channels interactively or type them into a file as shown below:

```
****define sender channel from NT to AS/400*****
def chl(NT1.QM400) chltype(sdr) trptype(tcp) conname(9.24.104.162) +
         xmitq(QM400)

****define receiver channel from AS/400 to NT*****
def chl(QM400.NT1) chltype(rcvr) trptype(tcp)

****define transmission queue for data going to AS/400****
def ql(QM400) usage(xmitq)

****define remote queue pointing to queue on AS/400****
def qr(J1) rname(J1) RQMNAME(QM400)
   ***Note - no xmitq named because same as RQMNAME***

****define local target queue on NT called N1****
def ql(N1)
```

Then create the objects with runmqscd using the file as input:

```
runmqsc  < mqscfil.txt
```

7.4.3 Sending and Receiving Messages

In this example, we send some messages from the Windows NT system to the AS/400 and vice versa.

Step 1. Start listeners and the channels in both systems:

AS/400: **STRMQMLSR**

STRMQMCHL CHLNAME(MQ400.NT1)

Windows NT: **runmqlsr -t tcp**

runmqchl -c NT1.MQ400 -m NT1

Step 2. Send messages from Windows NT to AS/400.

MQSeries provides the sample program AMQSPUT to put messages into a queue. It requires a queue name as input parameter. We specify the remote queue J1 which corresponds to the local queue J1 in the AS/400. Start the program and then type some messages. The example below shows the input in **bold**. To end the program press Enter.

```
amqsput J1
Sample AMQSPUT0 start
target queue is J1
1234 qwer ab

Sample AMQSPUT0 end
```

Step 3. Display the messages sent to AS/400 by keying:

WRKMQMMSG QNAME(J1)

Below is the output from the sample program AMQ1GBR4 (described in 6.4.2, “Browse Messages in a Queue” on page 127. Note that the browse program does not translate ASCII to EBCDIC as AMQ1GET4 does. Use WRKSPLF to browse the printout:

```
     Sample AMQ1GBR4 start
        Messages for J1
 00001 <èáëè (áëëåá        >
          no more messages
     Sample AMQ1GBR4 end
```

Step 4. Send messages from the AS/400 to the Windows NT system (the parameter MUST be 48 characters long)

CALL AMQ1PUT4 PARM('N1   ')

The AMQ1PUT program writes a log at AS/400 (WRKSPLF):

```
Sample AMQ1PUT4 start
Target queue is N1
Sample AMQ1PUT4 end
```

Step 5. Display the messages at the Windows NT machine:

amqbcg N1

The receiving end must do the EBCDIC/ASCII translation.

7.4.4 Monitoring Channels

You may want to know what happens to the channels while the two queue managers communicate. In the AS/400, you can use the commands:

- WRKMQMCHL (work with MQM channels)
- WRKMQMCHST (work with channel status)

WRKMQMCHL displays the run status of the channels. An example is shown in Figure 85.

```
                          Work with MQM Channels

Type options, press Enter.
  2=Change   3=Copy   4=Delete   5=Display   8=Work with Status   13=Ping
  14=Start   15=End   16=Reset   17=Resolve

Opt   Name                  Type         Transport    Status
__    NT1.QM400             *RCVR        *TCP         RUNNING
__    QM400.NT1             *SDR         *TCP         RUNNING









                                                                    More...
Parameters or command
===> ____________________________________________________________________________
F3=Exit   F4=Prompt      F5=Refresh   F6=Create   F9=Retrieve   F12=Cancel
F16=Repeat position to   F17=Position to          F21=Print
```

Figure 85. Work with MQM Channels

WRKMQMCHST is good for trouble shooting. It displays the sequence number for the last batch of messages on which both systems should agree. It also displays if messages cannot be delivered and (after pressing F11) the logical unit of work identifier for the last batch of messages.

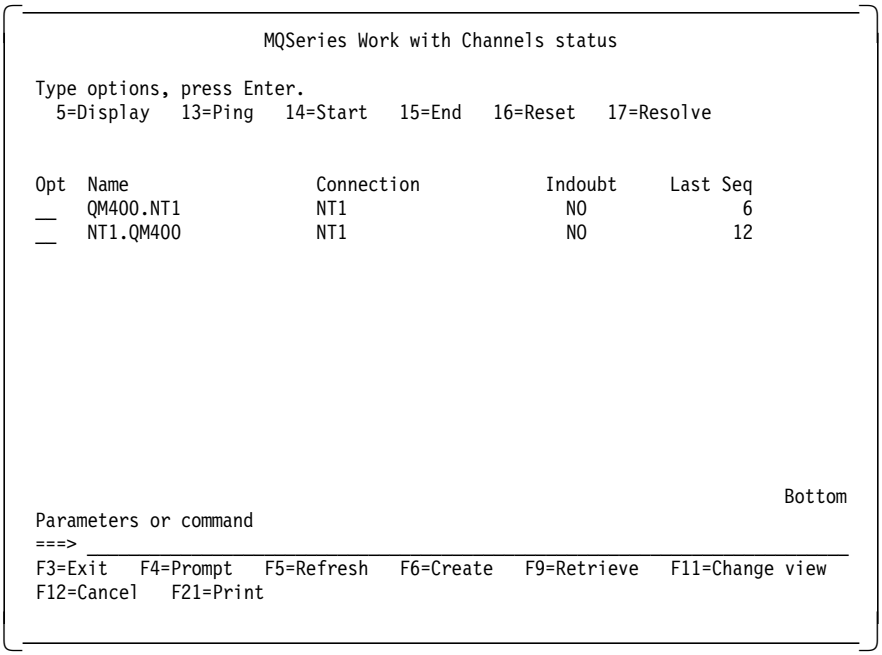


Figure 86. MQSeries Work with Channels Status

The channel status panel has 3 views. You can toggle between them using F11.

Be careful

Option 16 in front of the channel name resets the counter. Resetting this sequence number for a channel brings the sender/receiver channel pair out of sync, thus requiring resetting or restarting the counterparts in the other system.

To check if a channel pair is operational, you can *ping* it from the sender side. You can type 13 (ping option) in front of the MQ400.NT1 channel. You may prompt option 13 with F4; the command prompted is PNGMQMCHL (ping MQM channel). You must be able to ping the channel pair NT.QM400 from the NT side. The wellknown PING command verifies that TCP/IP is started at both sides. PNGMQMCHL verifies that MQ channels are operational.

Chapter 2, “MQSeries Overview” on page 7 has more details on defining MQ objects. Also take a look at *MQSeries for NT Quick Beginning*, GC33-1871.

Chapter 8. Operating MQSeries for AS/400

You should not need to start and stop MQ manager every day, and you should not need to IPL AS/400 every week.

You may have to stop applications in order to take a backup.

MQ is usually integrated in an application. So you should not have one schedule for backing up applications and another for MQ. MQ objects containing data (queues and channels) should be saved together with the files MQ serves. MQ uses journaling, so at least the transaction files ought to be under journaling.

8.1 Handling Journal Objects in Backup and Recovery

MQSeries for AS/400 uses journal management in order to keep queues, channels and messages persistent.

The correct handling of journals and journal receivers is the responsibility of the user. If journal receivers are not managed they will grow until all disk space is used up.

If journal management is totally new to you, please read 3.14, “Basic AS/400 Backup and Recovery” on page 76.

By the use of journal management (that contains database changes) you may also considerably minimize the need for a daily stop because of backups. Journals can change receivers and take a backup of journals and receivers on the fly.

We recommend this general procedure for handling journal objects. You may have other preferences and the publication *Backup and Recovery - Advanced*, SC41-4305 should be consulted. Use the following as a guideline:

1. Late in the evening you should change all receivers:

   ```
   CHGJRN xx  *GEN
   ```

   New receivers are created, the old ones are detached, the new ones are attached.

2. Take your necessary database backup. You may not need a file backup every day.

   The backup event is now logged in the attached database receiver.

3. Back up your MQ environment with RCDMQMIMG to the MQ journal.

© Copyright IBM Corp. 1998

The backup event is now logged in the attached MQ receiver.

4. Save the journals and journal receivers offline on tape.

   Always save BOTH journal and receivers at the same instant. Journals contain information about the attached receiver.

5. Delete journal receivers older than the recovery point. They are no longer needed.

6. Continue processing until next night.

Restore in Order

This is thge correct order for restoring objects associated with a journal:

1. Journals
2. Based-on physical files
3. Dependent logical files
4. Journal receivers

The scenario for a system restore is as follows:

1. Restore system, user profiles and configuration.
2. Prohibit users from getting started.
3. Restore journal receivers (only enough necessery for recovery).
4. Restore application database and code.
5. Restore IFS objects.
6. Restore authority.
7. Apply journal changes to database.
8. Adjust recovery point (if you don't have commit/rollback).
9. Test applications (make indexes if not journaled).
10. Have users start.

8.1.1 MQ Journals

MQ journals reside in the QUSRSYS library:

AMQAJRN controls persistency of local queues and their messages

AMQRJRN controls the channels

Note: Changes to the physical file AMQRFCD4 are journaled by AMQRJRN.

The corresponding journal receivers are in the QMQMDATA library:

AMQA000000 - AMQA999999
contains data for objects controlled by AMQAJRN

AMQR000000 - AMQR999999
contains data for objects controlled by AMQRJRN

If MQ objects are saved in journal receivers by RCDMQMIMG, you should know the name of the receiver that holds the recovery point. You should also know the names of the receivers that hold data after the recovery point. But don′t be intimidated by the job log. Before recovery takes place the system will investigate the chain of journal receivers. The first entry in each journal receiver is the name of the previous receiver. That is why there is an entry in the job log that AS/400 cannot find some old journal receiver. It is just a comment. It is not necessarily the case that the receiver is needed.

Remember: If not managed, journal receivers will continue to grow.

Two attributes of a journal can be changed to let OS/400 control the receivers:

- CHGJRN JRN(AMQAJRN) MNGRCV(*SYSTEM)

  When a threshold is reached, a new receiver is created and attached, and the old receiver is detached.

- CHGJRN JRN(AMQAJRN) DLTRCV(*YES)

  Old detached receivers are deleted.

If you manage the receivers yourself on a daily base, you will be better off. You will not risk having an MQ object image on a receiver that the system has deleted.

The AMQRJRN receivers cannot be deleted.

You cannot take a backup of a file, if a job has a lock on it. If you run TCP/IP the listener job has a lock on the file. If you run LU 6.2 (SNA) the SNA job started by the routing entry in your your communications subsystem has a lock on the file AMQRFCD4. This means that if your AMQRJRN receivers grow so much that you want to hold only one day′s traffic in the receivers, you must take a daily backup of the AMQRFCD4 file. And thus you have two choices:

- Run a program daily that stops the listener and, if you use it, the SNA job running AMQCRS6A.
- Stop MQ manager daily.

Unfortunately, there is no ENDMQMLSR command, nor end of the SNA program. You have to write it yourself.

This is our recommendation for keeping MQ journal receivers under control:

- CHGJRN JRN(AMQAJRN) JRNRCV(*GEN)
  /* at backup time (late evening) */
- CHGJRN JRN(AMQRJRN) JRNRCV(*GEN) /* new fresh receivers */
- Take down the TCP/IP listener, the SNA program or, at worst, end and quiesce MQ manager
- RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL)
  /* MQ objects saved in AMQAxxxxxx */
- SAVOBJ OBJ(AMQRFCD4) LIB(QMQMDATA) DEV(TAP01)
  OBJTYPE(*FILE)
- SAVOBJ OBJ(AMQ*) LIB(QUSRSYS) DEV(TAP01) OBJTYPE(*JRN)
- SAVOBJ OBJ(AMQ*) LIB(QMQMDATA) DEV(TAP01) OBJTYPE(*JRNRCV)
- DLTJRNRCV JRNRCV(QMQMDATA/AMQ*)
  /* all detached MQ receivers are deleted */
- SAVOBJ OBJ(AMQRJRN) LIB(QUSRSYS) DEV(TAP01) OBJTYPE(*JRN)
- STRMQMLSR, STRSBS of communication subsystem, or STRMQM

8.1.2 Recovering a Damaged MQ Object

This example illustrates the use of image logging:

RCDMQMIMG Record MQM object image

RCRMQMOBJ Re-create MQM object

1. Check if the attached receivers are not deleted or controlled by the system:

   WRKJRNA JRN(AMQAJRN)

2. Execute from the command line:

   CHGJRN JRN(AMQAJRN) JRNRCV(*GEN)

   ```
   Journal receiver AMQA000001 created in library QMQMDATA.
   Journal receivers AMQA000000 and *N detached.
   Sequence number not reset. First sequence number is 97.
   ```

3. Watch the job log (DSPJOBLOG) to see if a receiver (AMQA000000, for instance) has been detached and a new one (AMQ000001) has been created and attached.

   DSPJOBLOG

4. Continue keying and viewing the job log:

CRTMQMQ QNAME(TESTX) QTYPE(*LCL)
MQM queue created.
RCDMQMIMG OBJ(TESTX) OBJTYPE(*Q)
MQM object image recorded.

The image of queue TESTX must be recorded in AMQA000001. You may view the entries in the receiver with the command:

DSPJRN JRN(AMQAJRN)

Type option 5 for the journal entry MI as shown in Figure 87.

```
                          Display Journal Entries

Journal  . . . . . :   AMQAJRN         Library  . . . . . :   QUSRSYS

Type options, press Enter.
  5=Display entire entry


Opt    Sequence  Code  Type  Object      Library     Job         Time
 _           97   J     PR                          QPADEV0011   8:49:40
 _           98   J     JR   AMQA000001  QMQMDATA    QPADEV0011   8:49:40
 _           99   U     CO                          QPADEV0011   8:56:01
 _          100   U     QB                          QPADEV0011   8:56:01
 _          101   U     QA                          QPADEV0011   8:56:02
 _          102   U     CC                          QPADEV0011   8:56:02
 _          103   U     QM                          QPADEV0011   8:58:14
 _          104   U     MM                          QPADEV0011   8:58:14
 _          105   U     QM                          QPADEV0011   8:58:14
 5          106   U     MI                          QPADEV0011   8:58:14
 _          107   U     CL                          QPADEV0011   8:58:14
 _          108   U     CO                          QPADEV0010   9:06:06    +

F3=Exit   F12=Cancel
```

Figure 87. Display Journal Entries

The displayed entry (MI) is long. You may scroll it and view it in hexadecimal if you want.

5. Now work with the QMQMDATA library using STRPDM. Enter the library name QMQMDATA. This displays the panel shown in Figure 88 on page 158.
6. Find the userspace containing the MQ queue TESTX and delete it by typing option 4 in front of it.

   Press Enter on the confirmation panel. You get this confirmation message:

   Object TESTX in QMQMDATA type *USRSPC deleted.

```
                          Work with Objects Using PDM                 RALYAS4C

Library . . . . .   QMQMDATA__       Position to . . . . . .   __________
                                     Position to type  . . . . .   __________

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display       7=Rename
  8=Display description        9=Save       10=Restore      11=Move ...

Opt  Object      Type        Attribute   Text
__   QXSL00000   *USRSPC     SHMEM       XCS - Shared Storage
__   QXSL00001   *USRSPC     SHMEM       XCS - Shared Storage
__   QXSMEMTOP   *USRSPC     XSTROOT
4_   TESTX       *USRSPC                 Q TESTX      Storage




                                                                       Bottom
Parameters or command
===> ____________________________________________________________________________
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
```

Figure 88. Work with Objects Using PDM

7. At this point it is convenient to sign off and sign on again. MQ keeps information about its objects in the QTEMP library and in the activation group. So when you try to recover, MQ may not understand why everything has disappeared. RCLRSC reclaim resources can also be used.

8. Now recover the damaged object with the command:

   RCRMQMOBJ OBJ(TESTX) OBJTYPE(*Q)

   Watch in the job log. If you get a message that the queue TESTX is damaged (Object TESTX, type queue damaged), then try the command again. Eventually you get the messages below (and a lot more). When MQ has marked the object as damaged, you can re-create it.

   ```
   MQM object re-created - reapply authorities.
   MQM object recreated.
   ```

8.2 Channel Operations

When channels are started by the channel initiator they are easy to handle. The channel starts after the first message has arrived in the transmission queue (not before). In order to stop such a channel we recommend that you specify a disconnect timeout interval.

Note: Refer to Chapter 2, “MQSeries Overview” on page 7 for more information.

If you start channels manually, you may use the command STRMQMCHL. In this case you will probably specify a disconnect interval of zero, so that the channel will not stop when inactive.

Remember that the sender channel is the one in charge. From the sender side you initiate communication. At the receiver side the listener (TCP/IP or SNA) waits for data on the line and directs it to the receiver channel with the same name as the sender channel.

The panel displayed by the MQ CL command WRKMQMCHL is a good starting point for checking the channel status (Figure 89). The command WRKMQMCHST, work channel with status, is not; it is for trouble shooting.

```
                          Work with MQM Channels

Type options, press Enter.
  2=Change   3=Copy   4=Delete   5=Display   8=Work with Status   13=Ping
  14=Start   15=End   16=Reset   17=Resolve

Opt   Name                  Type         Transport    Status
__    CHANNEL.NTE.TO.AS1    *RCVR        *TCP         INACTIVE
__    CSQ1.TC.RALYAS4C      *RCVR        *TCP         INACTIVE
__    DIETER.QM400          *RCVR        *TCP         INACTIVE
__    ERICKA.400            *RCVR        *TCP         INACTIVE
__    NLRA134.TC.RALYAS4C   *RCVR        *TCP         INACTIVE
__    NT1.QM400             *RCVR        *TCP         INACTIVE
__    QM400.DIETER          *SDR         *TCP         INACTIVE
__    QM400.NT1             *SDR         *TCP         INACTIVE
__    RALYAS4C.TC.CSQ1      *SDR         *TCP         INACTIVE
__    RALYAS4C.TC.NLRA134   *SDR         *TCP         INACTIVE
__    RALYAS4C.TC.RS60001   *SDR         *TCP         INACTIVE
                                                                More...
Parameters or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt      F5=Refresh   F6=Create   F9=Retrieve   F12=Cancel
F16=Repeat position to   F17=Position to         F21=Print
```

Figure 89. Work with MQM Channels

The channel status can be one of the following:

Table 10. Channel Status	
BINDING	Channel is establishing a session and initial data exchange.
INACTIVE	Channel has ended processing normally or channel has never been started.
PAUSED	Channel is waiting for message-retry interval.
REQUESTING	Channel has been requested to start, but whether the channel has actually started is not known.
RETRY	Channel is waiting until time for next attempt.
RUNNING	Channel is transferring or is ready to transfer data.
STARTING	Channel is ready to begin negotiation will target MCA.
STOPPED	Channel has stopped because of an error.
STOPPING	Channel is in the process of closing.

If you have set a channel in STOPPED status, you must start it with STRMQMCHL.

Even a receiver channel that has been stopped (for some reason) must be started in order to go to INACTIVE status, so that BINDING and RUNNING can take place, initiated by the listener. Receiver channels usually go to INACTIVE status when MQ manager is started.

The channel status panel has three different views (toggled by F11). You need the information if there is a problem with the message traffic.

You can control the sequence number or the *logical unit of work ID* with the values for the channel at the remote MQ site. They should be the same. If they are not you can try to *Resolve*. Be very careful not to reset the counters (with option 16). Resetting the last sequence count (to zero) at one site, requires the other site to do the same, before traffic will flow again.

The Ping option (13) will execute the command: PNGMQMCHL, Ping MQM Channel. You may prompt it with option 13 and F4.

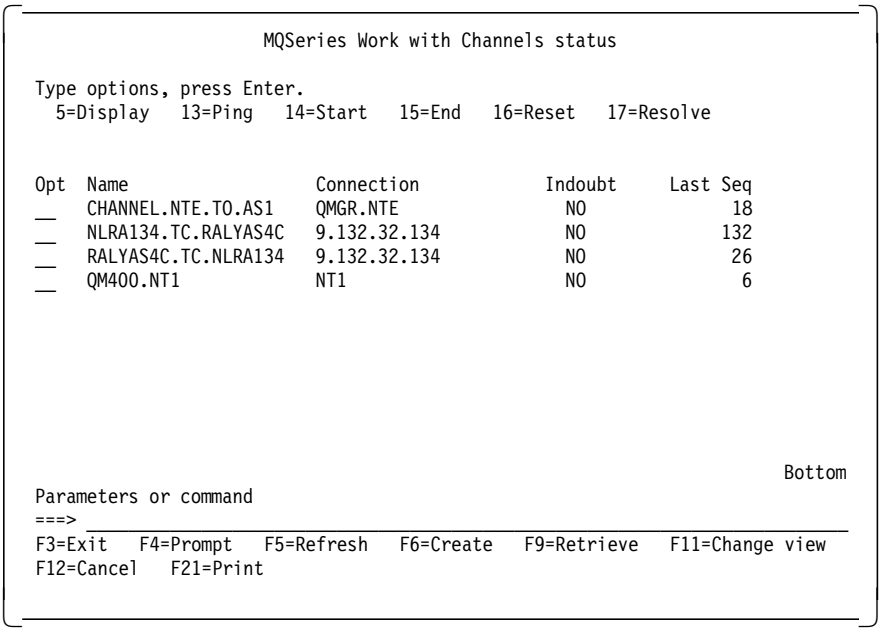


Figure 90. MQSeries Work with Channels Status

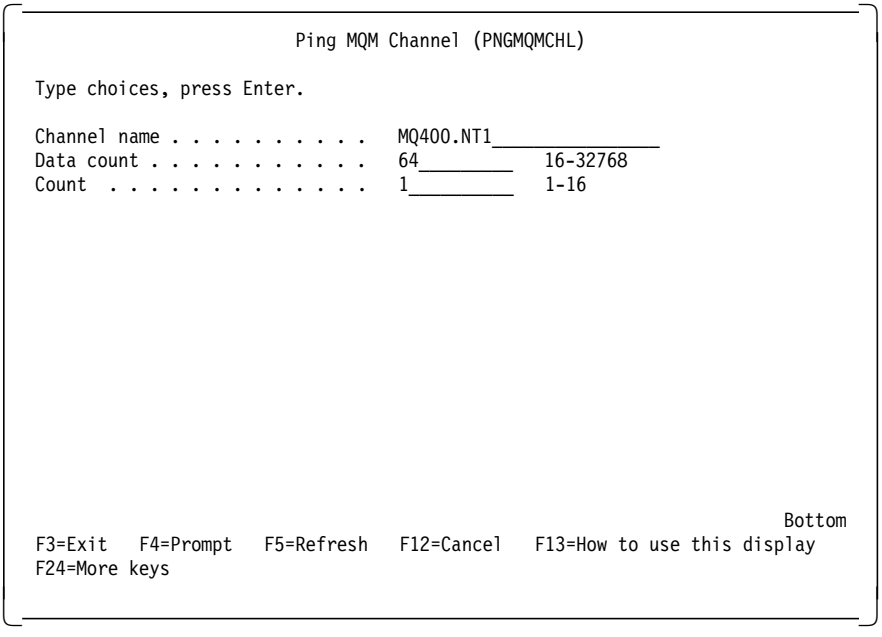


Figure 91. Ping MQM Channel (PNGMQMCHL)

The PNGMQMCHL will verify that the channel pairs (at this and the remote site) are working properly. PNGMQMCHL only works from the sending MQ manager.

Note: The TCP/IP ping command verifies that TCP/IP is operational.

There is also a PNGMQM (ping MQ manager) command. This MQM command tests a selected remote-queue manager by sending data as a special message and checking that the data is returned.

Restriction: PNGMQM only works correctly when issued from within the MQSeries for AS/400 administration utility (STRMQMADM). If you have issued this command directly from the command line do not continue using the command, press F3 or F12 to exit.

8.3 Starting MQ Operations

Problems when starting MQ are usually caused by problems that occurred when the MQM was stopped earlier.

If you cannot start the MQ manager because the previous stop left MQM jobs or objects in an unwanted state, stop the remaining MQ programs with the quiesce program AMQIQES4. We describe in the next section how to end the MQ manager.

You have to understand that a lot of MQ work is not under the direct control of the MQM. Some of these jobs are necessary for MQ to do its work for your application. Other types of MQ jobs are part of the application you have written (using MQ APIs).

You may have to execute some or all of the commands below to get your MQSeries environment running:

STRMQM start MQ manager

STRMQMLSR start listener (if TCP/IP)

STRSBS QCMN start comm subsystem (if SNA comm in QCMN)

STRMQMCHLI start channel initiator (if used)

STRMQMCHL start channels (if not started by channel initiator)

STRMQMCSVR start command server (if needed, remote control)

SBMJOB AMQSERVA
start trigger server or AMQSTRGA trigger monitor (for each monitored trigger queue)

Note: Not all STR commands have a corresponding END command; for example, there is no ENDMQMLSR (end listener).

While the queue manager is running some never ending jobs will start, and they do not all stop when MQ manager ends. If some of these jobs are active when the queue manager is started again you get a message. Don′t panic if you get MQ messages during startup. They may just be information telling you that some routine is already active.

Table 11 shows some processes that may be active in the subsystem QSYSWRK, QCMN (or QSNADS).

Table 11. MQSeries Processes	
AMQXIHK4	Storage monitor (never ending)
AMQALMP4	Check point process (running when MQM active)
AMQPCSVA	Command server, user controlled (STRMQMADM, STRMQMCSVR)
AMQMCPRA	Cache for Admin utility (never ending)
AMQCLMAA	TCP/IP listener (never ending)
AMQRIMNA	TCP/IP incoming channel initiator
AMQCCCLA	TCP/IP trigger monitor for sender MCA
AMQCRS6A	SNA LU 6.2 channel receiver
AMQRMCLA	Channel MCA program (one for each channel)

8.4 Stopping MQ Operations

You should generally let MQM and its associated processes run all through the week or even longer. You only need to stop the MQ manager:

- Before you install fixes for MQ
- Before you install *deferred* fixes to OS/400 (requires IPL)
- Before IPL
- Before running a system save in dedicated mode (only operator active)

In order to end MQ processes that do not stop with the ENDMQM command, we use three utilities:

1. **QMQM/AMQIQEJ4** Quiescing

   Ends all jobs running under *QMQM user profile*. This program also writes local MQ object information (RCDMQMIMG) to the journal receiver (AMQAnnnnnn) attached to the journal AMQAJRN.

2. **QMQM/AMQIQES4** Quiescing

Ends all jobs that are in the library list, except for the job executing the command. For the job AMQIQES4, take QMQM out of the library list.

3. **QMQM/AMQIQEM4** Quiescing

   To be run after AMQIQES4 to clean up MQ object cache storage.

To bring MQ jobs to a complete stop before a backup you have to execute the following commands:

ENDMQMCSVR	End command server (inhibit remote control).
*CHGMQMQ GETENBL(*NO)*	For triggering queues, ends application triggering.
*CHGMQMQ GETENBL(*YES)*	Reset.
*ENDMQMCHL xxx *CNTRLD*	Ends channel triggering in a controlled way.
*ENDMQM *CNTRLD*	Ends MQ manager in a controlled way, that is after all local API applications have finished.
CALL QMQM/AMQIQEJ4	Controlled quiescing, channels end before the storage monitor (AMQ6154 - means AMQIQEJ4 ran successfully).
CALL QMQM/AMQIQES4	No excuse quiescing, ends all remaining MQ jobs and some applications, and also disconnects MQ programmers that are logged on
CALL QMQM/AMQIQEM4	Clean up MQ cache.

Ending the queue manager with ENDMQM *IMMED is not necessary, if AMQIQES4 has been executed. You just get the message AMQ8142 (queue manager stopped) if you try.

You can see example programs that start and stop MQ operations in Appendix A, “Sample Programs” on page 187.

Note: If you use channel initiation, you don′t have to start any channels in your start MQ program. Without channel initiation, the sender channels must be started. If receiver channels have been stopped, they must be started in order to switch from *Stopped* to *Inactive*.

The CL program in A.8, “Starting Sender Channels” on page 192 has been used to start channels that have not been started by the channel initiatitor.

Only sender channels are started (type 1). By changing STRMQMCHL to ENDMQMCHL the program will stop channels.

Again if you trigger sender channels, and let the MCA (message channel agent) end them by a disconnect interval, you will not need this program.

Chapter 9. MQSeries Administration

There are several techniques for administrating an MQSeries for AS/400 installation. Here are some:

- MQ CL commands (like WRKMQMQ etc.)
- MQSC commands
- The Administration Utility (STRMQMADM)
- From a remote site (via STRMQMADM or MQSC commands to a queue)

9.1 CL Commands for MQ Administration

You can use these commands interactively, or you can put them into a source file and compile the source into a program.

The AS/400 professional will find MQ CL commands familiar and easy to use.

There is also a way to execute a source file with CL commands without having them compiled. You must then substitute the `PGM` and the `ENDPGM` with `// JOB` and `// END`, and then start execution with:

```
STRDBRDR FILE(QCLSRC) MBR(CLPGM1)
```

However, we don't recommend it. A compiled CL program is much easier and it can contain IF-THEN-ELSE logic.

To find the MQ CL commands type on a command line:

```
GO CMDMQM
```

This will display a menu that leads you to all MQM commands. You have five panels to scroll through. Figure 92 on page 168 shows one of them.

You can also type the beginning of a command followed by an asterisk (*), such as:

```
WRKMQM*
```

This will display all commands beginning with WRKMQM as you can see in Figure 93 on page 168.

Some commands require the name of the MQ manager to be keyed, others do not.

© Copyright IBM Corp. 1998

```
CMDMQM                          MQSeries Commands

Select one of the following:

  Queue Manager Commands
     1. Change Message Queue Manager                          CHGMQM
     2. Connect MQM                                           CCTMQM
     3. Create Message Queue Manager                          CRTMQM
     4. Delete Message Queue Manager                          DLTMQM
     5. Disconnect MQM                                        DSCMQM
     6. Display Message Queue Manager                         DSPMQM
     7. End Message Queue Manager                             ENDMQM
     8. Start Message Queue Manager                           STRMQM

  Command Server Commands
     9. Display MQM Command Server                            DSPMQMCSVR
    10. End MQM Command Server                                ENDMQMCSVR
    11. Start MQM Command Server                              STRMQMCSVR

                                                                More...
Selection or Command
===> ____________________________________________________________________
F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
```

Figure 92. MQSeries Commands

```
                                Select Command

Type options, press Enter.
  1=Select

Opt  Command     Library     Text
 _   WRKMQMCHL   QSYS        Work with MQM Channels
 _   WRKMQMCHST  QSYS        Work with MQM Channel Status
 _   WRKMQMMSG   QSYS        Work with MQM Messages
 _   WRKMQMPRC   QSYS        Work with MQM Processes
 _   WRKMQMQ     QSYS        Work with MQM Queues
 _   WRKMQMCHL   QMQM        Work with MQM Channels
 _   WRKMQMCHST  QMQM        Work with MQM Channel Status
 _   WRKMQMMSG   QMQM        Work with MQM Messages
 _   WRKMQMPRC   QMQM        Work with MQM Processes
 _   WRKMQMQ     QMQM        Work with MQM Queues


                                                              Bottom
Parameters or command
===> ____________________________________________________________________
F3=Exit      F4=Prompt    F5=Refresh    F9=Retrieve    F11=Display names only
F12=Cancel   F16=Repeat position to    F17=Position to    F24=More keys
```

Figure 93. Select Commands

Any errors or comfirmations are shown in the job log. Get to it in either of two ways:

1. Type DSPJOBLOG, press Enter, followed by F10, and PgDn.
2. Move the cursor to the error message below the command line, press F1 and F10 (function display messages in job log).

9.2 Use of MQSC Commands

The beauty of MQSC commands is that they are the same on all (25) MQ platforms. The MQ professional will find these commands familiar.

In an AS/400, commands cannot be executed interactively. They must be coded in a text member (script file), and then executed through the AS/400 command STRMQMMQSC. The member AMQSCOMA in the file QMQMSAMP/QMQSC is an example. AMQSCOMA will create the default definitions for MQSeries for AS/400. The very same definition can also be created with the CL program QMQM/AMQSDEF4 (source member is in QMQMSAMP/QCLSRC):

```
STRMQMMQSC SRCMBR(AMQSCOMA4) OPTION(*RUN)
```

The option can be *RUN or *VERIFY.

Executing the MQSC commands produces a listing. If you haven't started a printer against your output queue, you can display the output on a 5250 session with the Work with Spooled Files command:

```
WRKSPLF
```

9.3 MQSeries for AS/400 Administration Utility

With this interactive routine you can also perform administration jobs, such as create, delete and display MQ objects), and not only at the AS/400 where you execute the command, but also at remote sites, such as another AS/400, a Windows NT or MVS system.

The user ID (the AS/400 user profile) that you intend to operate under using this administration utility must first be initiated by calling the program AMQSADM4 with the user profile as parameter, for example:

```
CALL QMQM/AMQSADM4 MQADM
```

You may initiate more than one user profile.

If you do remote administration, the user ID must be defined at the remote site.

You start the administration routine with the command:

```
STRMQMADM
```

The F4 prompt shows the options in the panel shown in Figure 94:

```
                      Start MQM Administrator (STRMQMADM)

 Type choices, press Enter.

 Group name . . . . . . . . . . . GRPNAME        *ALL_____
 Message Queue Manager name . . . MQMNAME        *ALL_____________________
 ___________________
 Start Queue Manager  . . . . . . STRMQM         *YES
 Start Command Server . . . . . . STRCSVR        *YES

                                                                      Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 94. Start MQM Administrator

Accept the defaults. This will ensure that the command server (AMQPCSVA) is started in the subsystem QSYSWRK.

The first time you execute the command there will be no group name. At the first screen you must supply the name of your local MQ manager and a group name.

Specify a group name of your choice. Do this by pressing F6. To this group you add MQ managers that you intend to control (Figure 96 on page 171).

Before trying to administer a remote site, be sure that the necessary queues and channels have been set up. You can see an example of an AS/400 connected to a Windows NT system in Chapter 7, “AS/400 Communicating with Other Systems” on page 141.

The following panels show you what it is all about. When you administer remote sites you will have to wait for the replies to be sent to you.

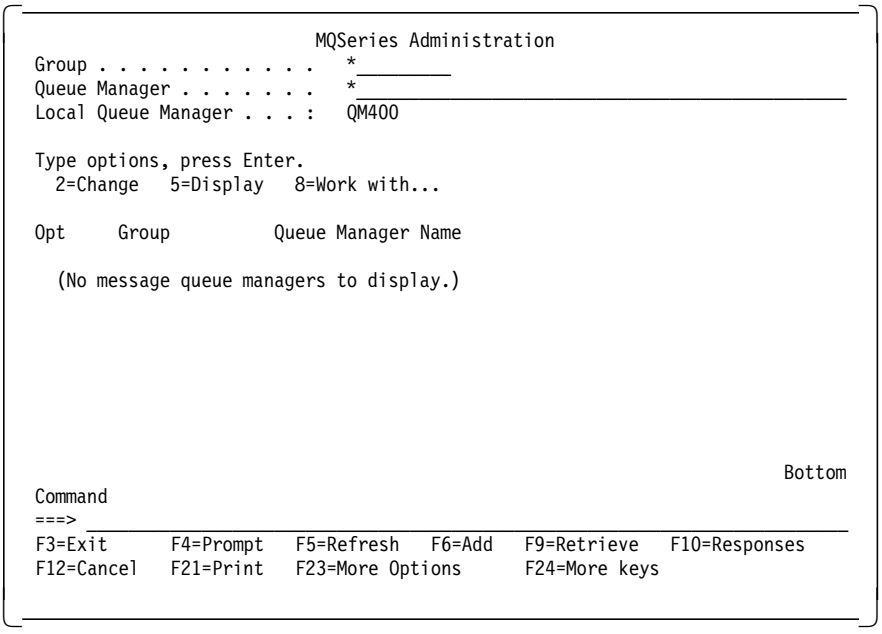


Figure 95. MQSeries Administration

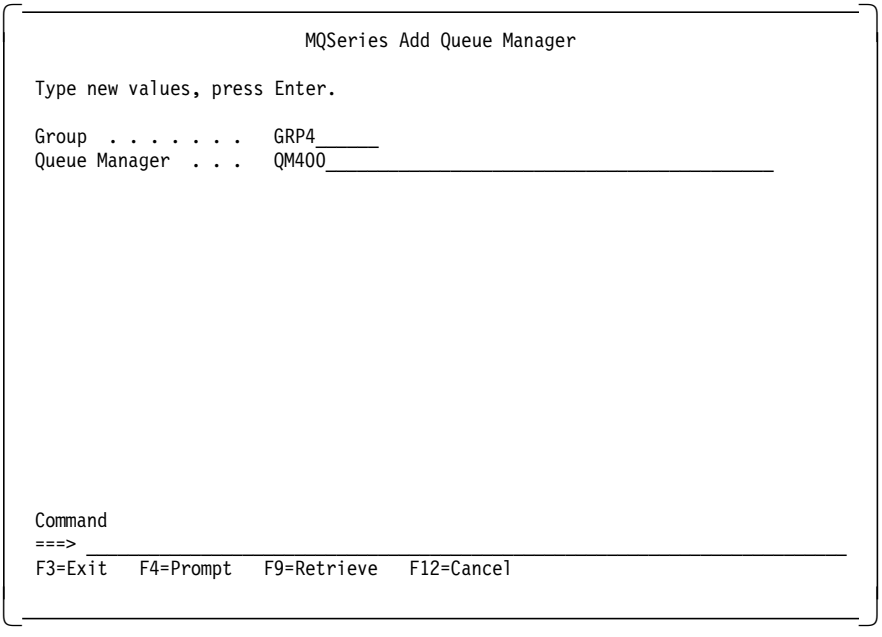


Figure 96. Add Queue Manager Panel

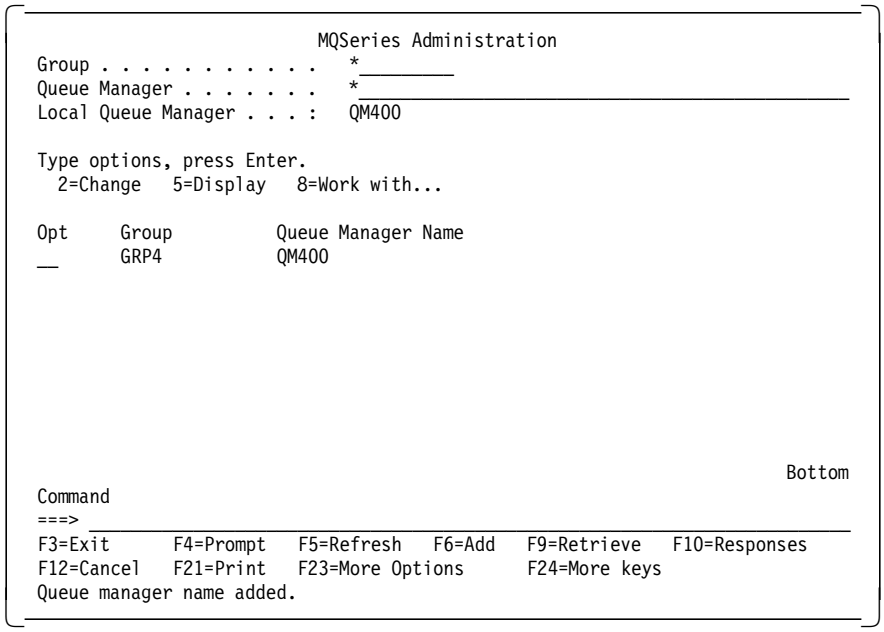


Figure 97. MQSeries Administration

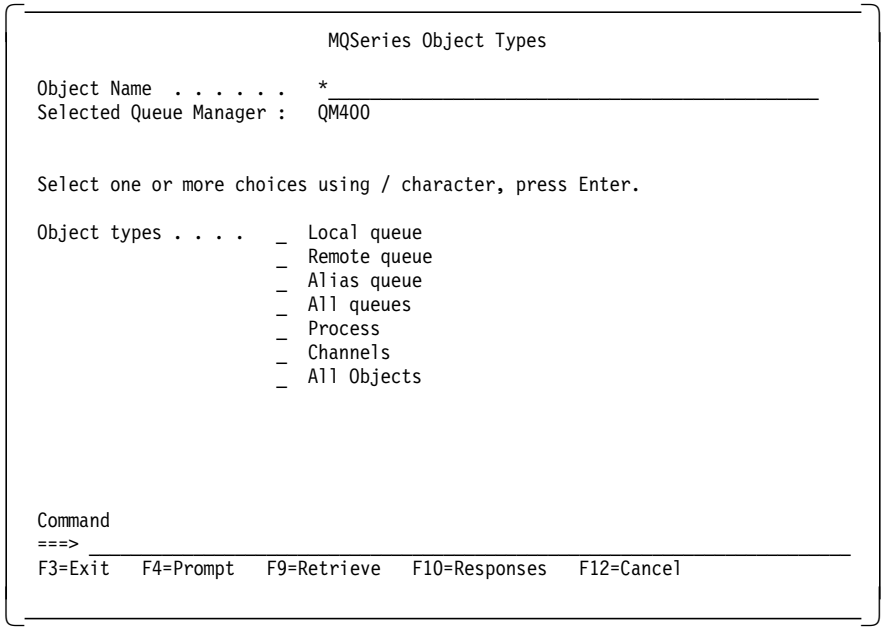


Figure 98. MQSeries Object Types

```
                          MQSeries Work with Objects

Queue Manager  . . . . . :   QM400

Type options, press Enter.
  1=Create   2=Change   3=Copy   4=Delete   5=Display
  8=Work with Messages...

Opt     Type     Sub        Object Name
__      ____     _____      ____________________________________________
__      *QUE     *ALS       SYSTEM.DEFAULT.ALIAS.QUEUE
__      *QUE     *LCL       SYSTEM.ADMIN.AMQMDATA.QUEUE
__      *QUE     *LCL       SYSTEM.ADMIN.CHANNEL.EVENT
__      *QUE     *LCL       SYSTEM.ADMIN.COMMAND.QUEUE
__      *QUE     *LCL       SYSTEM.ADMIN.EXCEPTION.QUEUE
__      *QUE     *LCL       SYSTEM.ADMIN.MQMLIST.MQADM
__      *QUE     *LCL       SYSTEM.ADMIN.PERFM.EVENT
__      *QUE     *LCL       SYSTEM.ADMIN.QMGR.EVENT
                                                                    More...
Command
===> ____________________________________________________________________
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Responses   F12=Cancel
F20=Nondisplay instructions/keys   F23=More Options   F24=More keys
```

Figure 99. MQSeries Display Queue

```
                              Display MQM Queue

Queue name . . . . . :   TEST2

Queue type . . . . . :   *LCL
Text 'description' . . . . . :

Put enabled  . . . . . :   *YES
Default message priority . . . :   0
Default message persistence  . :   *NO
Process name . . . . . :   PROC2

Triggering enabled . . . . . :   *YES
Get enabled  . . . . . :   *YES
Sharing enabled  . . . . . :   *YES
Default share option . . . . . :   *YES
Message delivery sequence  . . :   *PTY
Harden backout count . . . . . :   *NO
                                                                    More...



F3=Exit   F12=Cancel   F21=Print
```

Figure 100. MQSeries Work with Objects

```
                          Work with MQM Queues

Type options, press Enter.
  2=Change   3=Copy    4=Delete   5=Display   6=Clear   14=Display authority
  15=Grant authority   16=Revoke authority

Opt  Name                                          Type  Text
__   SYSTEM.ADMIN.AMQMDATA.QUEUE                   *LCL  System Admin Appl
__   SYSTEM.ADMIN.CHANNEL.EVENT                    *LCL  MQSeries channel
__   SYSTEM.ADMIN.COMMAND.QUEUE                    *LCL  MQSeries administ
__   SYSTEM.ADMIN.EXCEPTION.QUEUE                  *LCL  System admin appl
__   SYSTEM.ADMIN.MQMLIST.MQADM                    *LCL  Queue managers ma
__   SYSTEM.ADMIN.PERFM.EVENT                      *LCL  MQSeries performa
__   SYSTEM.ADMIN.QMGR.EVENT                       *LCL  MQSeries Queue Ma
__   SYSTEM.ADMIN.REPLYQ.MQADM                     *LCL  System admin appl
__   SYSTEM.ADMIN.RESPQ.MQADM                      *LCL  System admin appl
__   SYSTEM.CHANNEL.INITQ                          *LCL  MQSeries Channel
__   SYSTEM.CHANNEL.SYNCQ                          *LCL  MQSeries Channel
                                                                   More...
Parameters for options 2, 3, 5, 14, 15, 16 or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt      F5=Refresh   F6=Create   F9=Retrieve   F12=Cancel
F16=Repeat position to   F17=Position to          F20=Right     F21=Print
```

Figure 101. Work with MQM Queues

```
                          MQSeries Administration
Group . . . . . . . . . . .   *________
Queue Manager . . . . . . .   *____________________________________________
Local Queue Manager . . . :   QM400

Type options, press Enter.
  2=Change   5=Display   8=Work with...

Opt     Group          Queue Manager Name
__      GRP4           NT1
__      GRP4           QM400









                                                                     Bottom
Command
===> ____________________________________________________________________
F3=Exit      F4=Prompt   F5=Refresh   F6=Add   F9=Retrieve   F10=Responses
F12=Cancel   F21=Print   F23=More Options      F24=More keys
```

Figure 102. MQSeries Administration

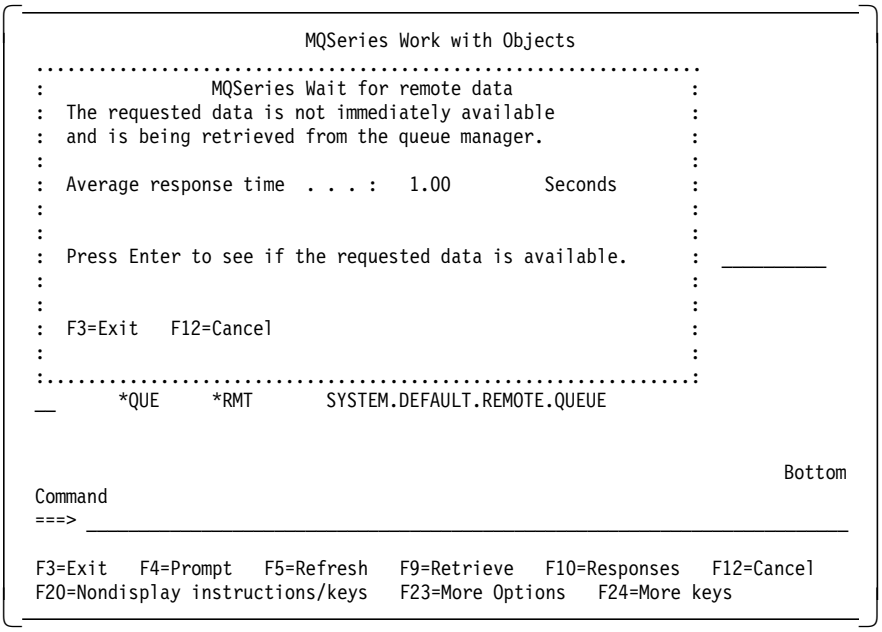


Figure 103. MQSeries Wait for Remote Data

9.4 Administering an AS/400 from a Remote Site

The installation of a queue manager can be administered from another AS/400 running STRMQMADM, and also from a remote user entering MQSC commands related to this MQ manager. The remote users must be known to the AS/400 and have the necessary authorization.

In this example, the command from the remote Windows NT user (at MQM NT1) WACKEROW was passed to the SYSTEM.DEAD.LETTER.QUEUE because AS/400 did not have his user profile.

Displaying a message can be done with either STRMQMADM or DSPMQMMSG. The message in Figure 104 on page 176 and Figure 105 on page 176 reads: `Display the local queues beginning with J at MQM MQ400`. The message `dis ql J*` is sent to MQM QM400.

The message contains a kind of PCF message (an escape PCF message). Many PCF messages have formats that cannot easily be read.

PCF is a format used to send commands across MQ managers.

```
                          Display MQM Message Data

Queue name . . . . . :   SYSTEM.DEAD.LETTER.QUEUE
Date  . . . . :   19980501              Type  . . . . :   REQUEST
Time  . . . . :   18245289              Format  . . . :   MQDEAD
Userid  . . . :   WACKEROW
Offset  Hexadecimal                          Text
0000    C4D3C840 00000001 000007F3 C1D4D84B   <DLH .......3AMQ.>
0010    F1F9F9F8 F0F5F0F1 F2F2F3F4 F0F2F7F8   <1998050122340278>
0020    40404040 40404040 40404040 40404040   <                >
0030    40404040 40404040 40404040 D5E3F140   <            NT1 >
0040    40404040 40404040 40404040 40404040   <                >
0050    40404040 40404040 40404040 40404040   <                >
0060    40404040 40404040 40404040 00000111   <            ....>
0070    00000025 D4D8C1C4 D4C9D540 00000008   <....MQADMIN ....>
0080    C1D4D8D7 C3E2E5C1 4040D8D4 D8D44040   <AMQPCSVA  QMQM  >
0090    40404040 F0F6F9F9 F2F50000 F1F9F9F8   <    069925..1998>
00A0    F0F5F0F1 F1F8F2F4 F5F2F0F2 00000001   <050118245202....>
00B0    00000024 00000001 00000026 00000001   <.................>
00C0    00000001 00000000 00000000 00000002   <.................>
00D0    00000003 00000010 000003F9 00000001   <...........9....>
                                                                  More...
F3=Exit   F12=Cancel   F21=Print
```

Figure 104. Display MQM Message Data

```
                          Display MQM Message Data

Queue name . . . . . :   SYSTEM.DEAD.LETTER.QUEUE
Date  . . . . :   19980501              Type  . . . . :   REQUEST
Time  . . . . :   18245289              Format  . . . :   MQDEAD
Userid  . . . :   WACKEROW
Offset  Hexadecimal                          Text
00E0    00000004 00000020 00000BC6 00000000   <...........F....>
00F0    0000000A 8489A240 98934DD1 5C5D0000   <....dis ql(J*)..>









                                                                  Bottom
F3=Exit   F12=Cancel   F21=Print
```

Figure 105. Display MQM Message Data

Chapter 10. Service and Trace

The standard AS/400 way of debugging, servicing and tracing programs and jobs also applies to MQSeries for AS/400, and applications written using MQ APIs (MQI).

In this chapter we explain the use of the specific MQ service and trace commands. They are very similar to the standard AS/400 commands, and they can give you additional information about MQ jobs.

Some jobs that cause problems may only stay active for a short time. This is annoying when you want to trace, because for a trace you need to know the full job name, and before you have executed the command WRKACTJOB (work active job) to display the job name, it is gone.

The key to success is to hold them on the job queue before they start executing.

In the following example, we will show how to trace the MQ command server. However, it could as well be one of your own programs, for instance a trigger monitor program.

In our example, we will have to hold the job queue QSYSNOMAX, because the command processor is scheduled to the subsystem QSYSWRK via this QSYSNOMAX.

When the job queue is held, you can submit the job to the job queue. The job will be allocated, but will not start executing. You can then display the job name, and start the trace commands and maybe change the job attributes to force a job log.

If you cannot finish your input before the job times out, you can schedule the job to start at a certain time instead. You can also prepare a CL program with the neccessary commands. Refer to A.1, “SWK - Display Active Jobs in Subsystem QSYSWRK” on page 187 for an example.

You need to be signed on at two AS/400 sessions as MQ administrator or someone else with the authority to do service.

In the first session, hold the job queue and submit the job:

```
HLDJOBQ JOBQ(QSYSNOMAX)
STRMQMCSVR MQMNAME(QM400)
```

© Copyright IBM Corp. 1998

```
                              Work with Job Queue

Queue:   QSYSNOMAX      Library:   QSYS           Status:   HLD/SBS

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release

Opt     Job            User           Number     Priority     Status
 _      AMQPCSVA       QMQM           069910        5          RLS

                                                                      Bottom
Parameters for options 2, 3 or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt   F6=Submit job   F12=Cancel   F21=Subsystem
F22=Work with job schedule entries    F24=More keys
```

Figure 106. Work with Job Queue

```
                              Work with Active Jobs                   RALYAS4C
                                                              05/01/98  10:49:50
CPU %:    3.5     Elapsed time:   01:32:05     Active jobs:   123

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...

Opt  Subsystem/Job  User        Type  CPU %  Function        Status
__      PRT01       QSPLJOB     WTR      .0                  EVTW
__   QSYSWRK        QSYS        SBS      .0                  DEQW
__     AMQALMP4     QMQM        BCH      .0  PGM-AMQALMP4    DEQW
__     AMQMCPRA     QMQM        BCH      .0  PGM-AMQMCPRA    DEQW
__     AMQPCSVA     QMQM        BCH      .0  PGM-AMQPCSVA    RUN
__     QAPPCIPX     QSYS        BCH      .0                  TIMW
__     QAPPCTCP     QSYS        BCH      .0  PGM-QZPAIJOB    TIMW
__     QCQEPMON     QSVMSS      BCH      .0  PGM-QCQEPMON    MSGW
__     QCQRCVDS     QSVMSS      BCH      .0  PGM-QCQAPDRM    MSGW
                                                                    More...
Parameters or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt        F5=Refresh   F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F14=Include   F24=More keys
```

Figure 107. Work with Active Jobs

In the second session enter (hurry before the job times out on the job queue):

```
WRKJOBQ JOBQ(QSYSNOMAX)
```

Note the job number in the panel shown in Figure 106 on page 178.

Then enter:

```
CHGJOB JOB(069910/QMQM/AMQPCSVA) LOG(*SAME *SAME *SECLVL) LOGCLPGM(*YES)
STRMQMSRV JOB(069910/QMQM/AMQPCSVA)
RLSJOBQ JOBQ(QSYSNOMAX)
```

Now the job starts running. Watch it with this command:

```
WRKACTJOB SBS(QSYSWRK)
```

When it has finished executing or shows the behavior, you want to trace enter:

```
TRCMQM SET(*OFF)
ENDMQMSRV
```

You can now read the output with the command WRKSPLF if it is not already directed to a printer. The trace listing is shown in Figure 110 on page 181.

The job log is viewed from the outq QEZJOBLOG, but only after the job has ended.

```
WRKOUTQ OUTQ(QEZJOBLOG)
```

Figure 108 on page 180 shows the panel. When the job is under service the job log will contain additional information.

Page forward, toggle the view with F11, and type option 5 (display) in front of the job name. Then page forward through the log shown in Figure 109 on page 180.

If you suspect the problem to be caused by a misbehavior of MQSeries contact IBM Service via your ECS line.

```
                          Work with Output Queue

Queue:   QEZJOBLOG      Library:   QUSRSYS        Status:   RLS

Type options, press Enter.
  1=Send   2=Change   3=Hold   4=Delete   5=Display   6=Release   7=Messages
  8=Attributes        9=Work with printing status

Opt  File        File Nbr  Job         User        Number  Date      Time
 _   QPJOBLOG         1    AMQPCSVA    QMQM        069906  05/01/98  10:28:08
 _   QPJOBLOG         1    AMQPCSVA    QMQM        069910  05/01/98  10:50:25
 _   QPJOBLOG         2    QTRTD41458  QTCP        069912  05/01/98  11:01:26
 _   QPJOBLOG         1    QTRTD47523  QTCP        069916  05/01/98  11:05:05

                                                                        Bottom
Parameters for options 1, 2, 3 or command
===> ____________________________________________________________________________
F3=Exit   F11=View 1   F12=Cancel   F20=Writers   F22=Printers
F24=More keys
```

Figure 108. Work with Output Queue

```
                           Display Spooled File
File  . . . . . :   QPJOBLOG                        Page/Line   1/53
Control . . . . .   W37                             Columns     37 - 114
Find  . . . . . .
...4....+....5....+....6....+....7....+....8....+....9....+....0....+....1....
 To procedure  . . . . . . . :   rrxGetNextChannelDef
 Statement . . . . . . . . . :   38           *PRCLT
 Message . . . . :   Pointer not set for location referenced.
 Cause . . . . . :   A pointer was used, either directly or as a basing
   pointer, that has not been set to an address.
0    05/02/98   20:08:45   AMQXLIB4      QMQM        *STMT    AMQXLIB4     QM
 From module . . . . . . . . :   AMQXEID4
 From procedure  . . . . . . :   xcsDisplayMessage
            Job Log                              RALYAS4C 05/02/98 20:08:57
MQPCSVA        User  . . . . . . . :   QMQM         Number . . . . . . . . . . .
DFTJOBD        Library . . . . . . :   QGPL
EV   DATE       TIME       FROM PGM       LIBRARY     INST     TO PGM       LI
 Statement . . . . . . . . . :   48
 To module . . . . . . . . . :   AMQXFDC4
 To procedure  . . . . . . . :   xcsFFST
 Statement . . . . . . . . . :   16
                                                                       More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

Figure 109. Display Spooled File

```
AMQPCSVA  QMQM      06991O
OOO > xcsQueryDateTime Sep  8 1997  V4R2MO
OOO < xcsQueryDateTime, ret OOOOOOOO
xtrInitTrace 8 xOOOO6136
Stopping early trace 1998-O5-O1  1O.49.58 system time
xtrInitTrace 1O xOOOO6134
Trace continues in buffer
**************************************************
xtrInitTrace 11 xOOOO6138
Resuming MQI trace 1998-O5-O1  1O.49.58 system time
**************************************************
OOO < xtrInitTrace, ret OOOOOOOO
OOO > xllInitESems Sep  8 1997  V4R2MO
OO1 .> xcsConnectSharedMem Sep  8 1997  V4R2MO
OO1 .< xcsConnectSharedMem, ret OOOOOOOO
OO1 .> xcsCreateSharedMemSet Sep  8 1997  V4R2MO
OO2 ..> xstDeqPtr Sep  8 1997  V4R2MO
OO2 ..< xstDeqPtr, ret OOOOOOOO
OO2 ..> xcsConnectSharedMem Sep  8 1997  V4R2MO
OO3 ...> xstValidateHSMS Sep  8 1997  V4R2MO
OO3 ...< xstValidateHSMS, ret OOOOOOOO
OO2 ..< xcsConnectSharedMem, ret OOOOOOOO
OO2 ..> xcsAllocateMemBlock Sep  8 1997  V4R2MO
OO3 ...> xstValidateHSMS Sep  8 1997  V4R2MO
OO3 ...< xstValidateHSMS, ret OOOOOOOO
OO3 ...> xstDeqPtr Sep  8 1997  V4R2MO
OO3 ...< xstDeqPtr, ret OOOOOOOO
OO2 ..< xcsAllocateMemBlock, ret OOOOOOOO
OO1 .< xcsCreateSharedMemSet, ret OOOOOOOO
OOO < xllInitESems, ret OOOOOOOO
OOO > xcsConnectSharedMem Sep  8 1997  V4R2MO
OOO < xcsConnectSharedMem, ret OOOOOOOO
OOO > xcsRequestMutexSem Sep  8 1997  V4R2MO
OOO < xcsRequestMutexSem, ret OOOOOOOO
OOO > xcsReleaseMutexSem Sep  8 1997  V4R2MO
OOO < xcsReleaseMutexSem, ret OOOOOOOO
OOO > xcsDisconnectSharedMem Sep  8 1997  V4R2MO
OO1 .> xstValidateHSMS Sep  8 1997  V4R2MO
OO1 .< xstValidateHSMS, ret OOOOOOOO
OOO <!xcsDisconnectSharedMem, ret OO8O6043 xecS_I_NO_OTHER_CONNS
OOO < xcsInitialize, ret OOOOOOOO
OOO > pcmMain Sep  8 1997  V4R2MO
OO1 .> xcsConnectSharedMem Sep  8 1997  V4R2MO
OO1 .< xcsConnectSharedMem, ret OOOOOOOO
OO1 .> xcsCreateSharedMemSet Sep  8 1997  V4R2MO
OO1 .<!xcsCreateSharedMemSet, ret 2O8O6O38 xecS_E_SET_EXISTS
OO1 .> xcsConnectSharedMem Sep  8 1997  V4R2MO
OO1 .< xcsConnectSharedMem, ret OOOOOOOO
OO1 .> xcsRequestMutexSem Sep  8 1997  V4R2MO
OO1 .< xcsRequestMutexSem, ret OOOOOOOO
OO1 .> xcsVerifyVars Sep  8 1997  V4R2MO
OO1 .< xcsVerifyVars, ret OOOOOOOO
OO1 .> xcsInitialize Sep  8 1997  V4R2MO
OO1 .<!xcsInitialize, ret OO8O6O66 xecI_I_ALREADY_INIT
OO1 .> zapMQCONN Jan 14 1998  V4R2MO
_________
MQCONN >>
name:
 D8D4F4F0 F0OO4B4O 4O4O4B4O 4B4O4B4O  ]QM4OO..   . . . ]
 4B4O4B4O 4O4O4B4O 4B4O4B4O C1D3C3D6  ]. .   . . . ALCO]
 C2D14O4O 4O4O4O5C C3D4C44O 4O4B4O4B  ]BJ     *CMD  . .]
hconn         : Output Parm
```

Figure 110. MQ Trace Listing

10.1 Before Contacting IBM Service

Before contacting IBM Service, check that you have installed the latest fixes. This Internet page has the latest information:

http://as400service.rochester.ibm.com/

- Select **Technical Information database**.
- Select **Preventive Service Planning (PSP)**.
- Click on **Search**.
- Enter 5769MQ1 in the search field.
- Select **FIX SUMMARY LISTING FOR VERSION 4 RELEASE 2.0**.
- Scroll down to the information about MQSeries for AS/400.

Figure 111 is an example of such a PTF listing.

```
            PRODUCT NAME: 5769MQ1 - V4 R2.0 - MQSERIES FOR OS/400


|PTF/    |     | AVAIL. |                                          |REPLACED |
|FIX #   |PKG# |  DATE  |ABSTRACT                                  |   BY    |
-------------------------------------------------------------------------------
SF43258  8028  01/31/98 MQM400-MSGMCH3401-MSGRC20806058 AMQIQEM4  SF44198
                        DELETES QXSMEMTOP
SF43307  8028  01/31/98 MQM400 IMPROVE MIGRATION OF THE           SF44856
                        SYNCHRONIZATION FILE
                        MQM400 MIGRATION OF 3.6 CHANNEL
                        DEFINTION FILE
SF43461  8028  01/31/98 MQM400 MIGRATION OF MQSERIES CHANNEL      SF44856
                        DEFINITION FILE
SF44104  8041  02/16/98 MQM400 SERVICE UPDATES TO MQSERIES CPP
SF44198  8041  02/16/98 MQM400 UPDATE PROGRAM AMQIQEU4, SHOULD
                        BE DOMAIN *USER
                        MQM400-MSGCPF1321 QUIESCE ENDJOB
                        AMQALMP4 FAILS AFTER ENDMQM
                        MQM400 SERVICE UPDATES TO MQSERIES MCS
SF44856  8041  02/16/98 MQM400 PROGRAM CHECK IN MCA RECEIVER JOB
                        MQM400 SERVICE UPDATES TO MQSERIES MCA &
                        MIGRATE TOOLS
```

Figure 111. PTF Listing

```
                    Display Program Temporary Fix (DSPPTF)

Type choices, press Enter.

Product  . . . . . .  5769mq1       F4 for list
PTF numbers to select  . . . . .  *ALL____      Character value, *ALL...
Release  . . . . . .  *ALL__        *ALL, VxRxMx
Cover letter only  . . . . . .  *NO_          *NO, *YES
Output . . . . . .  *_______      *, *PRINT, *OUTFILE

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 112. Display Program Temporary Fix (DSPPTF)

```
                              Display PTF Status
                                                              System:   RALYAS4C
Product ID  . . . . . :   5769MQ1
IPL source  . . . . . :   ##MACH#B
Release of base option  . . . . . :   V4R2M0


Type options, press Enter.
  5=Display PTF details   6=Print cover letter   8=Display cover letter

     PTF                                                      IPL
Opt  ID       Status                                        Action
 _   SF45304  Temporarily applied                           None
 _   SF45237  Temporarily applied                           None
 _   SF44856  Temporarily applied                           None
 _   SF44198  Temporarily applied                           None
 _   SF44104  Temporarily applied                           None
 _   SF43461  Permanently applied                           None
 _   SF43307  Superseded                                    None
 _   SF43258  Permanently applied                           None

                                                                      Bottom
F3=Exit   F11=Display alternate view   F17=Position to   F12=Cancel
```

Figure 113. Display PTF Status

Now check which PTFs you have loaded and applied using one of the following commands:

- Type DSPPTF press F4 (Figure 112 on page 183).
- Type GO PTF and then 5 (display) for 5769MQ (Figure 113 on page 183).

Compare your list against the list on the Internet. If you miss a PTF it must be ordered via the *ECS line*. For example, the AS/400 operator can get the PTF SF46965 with this command:

```
SNDPTFORD PTFID((SF46965 5769MQ1))
```

After you have received the PTF and the cover letter via the modem from your ECS line, you must load and apply the PTF with the following commands:

```
LODPTF LICPGM(5769MQ1) DEV(*SERVICE) SELECT(SF46965)
APYPTF LICPGM(5769MQ1) SELECT(SF46965)
```

Don′t forget to read the cover letter. Some PTFs require more action than load and apply.

Cover letters are also available from the Internet:

```
http://as400service.rochester.ibm.com/
```

- Select **Technical Information database**.
- Select **AS/400 PTF Cover Letters**.
- Scroll the items under R420.
- Select **FIX SUMMARY LISTING FOR VERSION 4 RELEASE 2.0**.
- Scroll down to the information about MQSeries for AS/400. .
- Select and read the current cover letters.

The *MQSeries for AS/400 Administration Guide* contains a chapter with valid questions to ask yourself while analyzing a problem.

We also recommend using the AS/400 menu:

```
GO PROBLEM
```

When the command WRKPRB (Work with Problems) is executed you will see all system messages that have required operator intervention.

Figure 114 on page 185 shows the ″Work with Problems″ panel that provides sevaral options to analyze the problem. Figure 115 on page 185 is an example of a Display Problem Details panel.

```
                              Work with Problems
                                                                 System:   RALYAS4C
Position to . . . . . .   _________      Problem ID

Type options, press Enter.
  2=Change   4=Delete   5=Display details    6=Print details
  8=Work with problem   9=Work with alerts   12=Enter text


Opt  Problem ID  Status         Problem Description
__   9812056400  OPENED         APPN session initiation attempt has failed.
__   9812056379  OPENED         APPN session initiation attempt has failed.
__   9812056043  OPENED         APPN session initiation attempt has failed.
__   9812055814  OPENED         APPN session initiation attempt has failed.
__   9812042883  READY          Software problem data for QTMHJOBS has been lo
__   9812042588  READY          *Attention*  Resource  removed or failed.
__   9812042584  READY          *Attention*  Resource  removed or failed.
__   9812042577  READY          *Attention*  Resource  removed or failed.
5_   9811976391  READY          Software problem data for AMQXLIB4 has been lo
__   9811937824  READY          *Attention*  Hardware service may be required.
                                                                        More...
F3=Exit      F5=Refresh   F6=Print list     F11=Display dates and times
F12=Cancel   F16=Report prepared problems   F24=More keys
```

Figure 114. Work with Problems

```
                           Display Problem Details
                                                                 System:   RALYAS4C
Problem ID . . . . . :   9811976391
Origin . . . . . :   USIBMRA.RALYAS4C
Current status . . . . . :   READY
Problem  . . . . . :   Software problem data for AMQXLIB4 has been logged
Refer to help text for addit

Problem message ID  . . . . . :   CPI93B9
Problem type  . . . . . :   Machine detected
Problem category  . . . . . :   *REPORT
Date and time detected  . . . . . :   04/29/98  22:15:01
System reference code . . . . . :   SRCB900FDC7









                                                                        More...
Press Enter to continue.

F3=Exit   F6=Display problem history   F11=Display APAR library   F12=Cancel
F16=Display spooled files   F20=Display submitted change requests
```

Figure 115. Display Problem Details

10.2 Contacting IBM Service

If you need support from IBM Service, you can report your problem, from the Work with Problem menu. Type 2 (report problem) and add some text.

Some more panels assist you in sending the report to IBM Service via the ECS line at your AS/400.

This way IBM gets your telephone number, your name and address, your CPU serial number, the error message and a lot more. Also, the IBM Service Engineer will get computer assisted support from a central service database, before he contacts you by phone.

```
                            Work with Problem
                                                              System:   RALYAS4C
 Problem ID . . . . . . . :   9812134224
 Origin . . . . . . . . . :   USIBMRA.RALYAS4C
 Current status . . . . . :   READY
 Problem  . . . . . . . . :   Software problem data for AMQXLIB4 has been logged
. Refer to help text for addit

 Select one of the following:.

      1. Analyze problem
      2. Report problem

      4. Verify problem corrected
      5. Answer problem

     20. Close problem

                                                                          More...
 Selection
      2_

 F3=Exit   F12=Cancel
```

Figure 116. Work with Problem

Appendix A. Sample Programs

A.1 SWK - Display Active Jobs in Subsystem QSYSWRK

In order to see which MQ jobs are active in subsystem QSYSWRK, you need to key all this:

```
WRKACTJOB SBS(QSYSWRK)
```

If you would like to key less, type these two instructions, to create a short command SWK that will do the same:

```
CRTDUPOBJ OBJ(WRKACTJOB) FROMLIB(QSYS)
          OBJTYPE(*CMD) TOLIB(QGPL)
          NEWOBJ(SWK)
CHGCMDDFT CMD(SWK) NEWDFT('SBS(QSYSWRK)')
```

A.2 A Simple File: FILA

This file structure is used in the program PGMA.

```
              R RECA
                NUMB            6  0
                NAME           15
                ADDR           15
```

A.3 A Simple RPG Program with Commit Control

This quick-and-dirty program simply adds three records to the file FILA opened under Commit Control.

```
     FFILA    O   E                    DISK         KCOMIT
     C                    Z-ADD1         NUMB
     C                    MOVEL'ATHUR '  NAME
     C                    MOVEL'TABLE '  ADDR
     C                    WRITERECA
     C                    Z-ADD2         NUMB
     C                    MOVEL'BERTHA'  NAME
     C                    MOVEL'CANNON'  ADDR
     C                    WRITERECA
     C                    Z-ADD3         NUMB
     C                    MOVEL'CAESAR'  NAME
     C                    MOVEL'RIVER '  ADDR
     C                    WRITERECA
     C                    SETON                     LR
```

© Copyright IBM Corp. 1998

A.4 DTA2 - Put Date and Time in a Data Area

This program puts the system date and time in an AS/400 data area. It is used in Chapter 6, “Running the Samples” on page 117 to illustrate that a program can be triggered, when the first message arrives in a queue.

Due to the design of the monitoring trigger programs AMQSERV4 and AMQSTRG4, the program must have a parameter of 684 characters.

This parameter contains information that a ″real″ application program will use, for instance, which MQ queue to continue working with.

```
PGM         PARM(&PARAM)
DCL         VAR(&PARAM) TYPE(*CHAR) LEN(684)
DCL         VAR(&TIM) TYPE(*CHAR) LEN(6)
DCL         VAR(&YER) TYPE(*CHAR) LEN(2)
DCL         VAR(&MON) TYPE(*CHAR) LEN(2)
DCL         VAR(&DAY) TYPE(*CHAR) LEN(2)
DCL         VAR(&CEN) TYPE(*CHAR) LEN(1)
DCL         VAR(&HUN) TYPE(*CHAR) LEN(2) VALUE('20')
MONMSG CPF0000
RTVSYSVAL   SYSVAL(QCENTURY) RTNVAR(&CEN)
IF          COND(&CEN = '0') THEN(CHGVAR VAR(&HUN) +
              VALUE('19'))
RTVSYSVAL   SYSVAL(QYEAR) RTNVAR(&YER)
RTVSYSVAL   SYSVAL(QMONTH) RTNVAR(&MON)
RTVSYSVAL   SYSVAL(QDAY) RTNVAR(&DAY)
RTVSYSVAL   SYSVAL(QTIME) RTNVAR(&TIM)
CRTDTAARA   DTAARA(MQPGM/DTA2) TYPE(*CHAR) LEN(100)
CHGDTAARA   DTAARA(MQPGM/DTA2 (1 30)) VALUE(&HUN *CAT +
              &YER *BCAT &MON *BCAT &DAY *BCAT '-' +
              *BCAT &TIM)
ENDPGM
```

A.5 TCJOB - Save Keying Time When Tracing a Job

This program will help you serve and trace a job, when you know the job number. The job name has been hardcoded.

```
PGM        PARM(&PRM)
DCL        VAR(&PRM) TYPE(*CHAR) LEN(32)
DCL        VAR(&JNR) TYPE(*CHAR) LEN(6)
MONMSG CPF0000
CHGVAR     VAR(&JNR) VALUE(%SST(&PRM 1 6))
STRMQMSRV  JOB(&JNR/QMQM/AMQPCSVA)
TRCMQM (*ON)
RLSJOBQ QSYSNOMAX
ENDPGM
```

A.6 Starting MQ Manager

This program has been used to start the MQ manager at a test installation.

```
            PGM

            DCL         VAR(&MSGID) TYPE(*CHAR) LEN(7)

            MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))

            STRMQM
            MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))
            RCVMSG      MSGTYPE(*LAST) RMV(*NO) MSGID(&MSGID)

            IF          COND(&MSGID *EQ 'AMQ8003') THEN(DO)

            STRMQMLSR
            MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))
 /*         IF          COND(&MSGID *EQ 'AMQ8021') THEN(DO) */

            STRMQMCHLI  QNAME(SYSTEM.CHANNEL.INITQ)
            MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))
            ENDDO

NORMAL:     SNDPGMMSG   MSG('IBM MQSERIES STARTED SUCCESFULLY')
            GOTO CMDLBL(END)

ERROR:      SNDPGMMSG   MSG('GENERAL ERROR OCCURED WHEN STARTING +
                         MQSERIES')

END:        ENDPGM
```

A.7 Ending MQ Manager

This program has been used to end the MQ manager at a test installation.

```
/* END ALL MQSERIES ASSOCIATED JOBS */

  LOOP:       CALL       PGM(QMQM/AMQIQES4)
              MONMSG     MSGID(AMQ6906) EXEC(GOTO CMDLBL(LOOP))
              MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))

/* RELEASE ASSOCIATED USER SPACES */

              CALL       PGM(QMQM/AMQIQEM4)
              MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))

  NORMAL:     SNDPGMMSG  MSG('IBM MQSERIES STOPPED SUCCESFULLY')
              GOTO CMDLBL(END)

  ERROR:      SNDPGMMSG  MSG('GENERAL ERROR OCCURED WHEN STOPPING +
                          MQSERIES')

  END:        ENDPGM
```

A.8 Starting Sender Channels

This program has been used to start sender channels not started by channel initiation:

```
            PGM

            DCLF       FILE(QMQMDATA/AMQRFCD4) ALWVARLEN(*YES)
            DCL        VAR(&CHANNAME) TYPE(*CHAR) LEN(20)
            DCL        VAR(&CHANTYPE) TYPE(*DEC) LEN(9)

READ:       RCVF
            MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(END))

            IF         COND(&CHANTYPE *EQ 1) THEN(GOTO CMDLBL(START))
            ELSE GOTO CMDLBL(READ)

START:      STRMQMCHL  CHLNAME(&CHANNAME)

            MONMSG     MSGID(CPF0001) EXEC(GOTO CMDLBL(READ))

            SNDPGMMSG  MSG('Channel' *BCAT &CHANNAME *BCAT 'started')
            GOTO       CMDLBL(READ)

END:
            ENDPGM
```

Appendix B. Special Notices

This publication is intended to help AS/400 and MQSeries specialists to install, maintain and administer MQSeries for AS/400. The information in this publication is not intended as the specification of any programming interfaces that are provided by MQSeries for AS/400 V4R2 and OS/400 V4R2. See the PUBLICATIONS section of the IBM Programming Announcement for MQSeries for AS/400 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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 Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each

© Copyright IBM Corp. 1998

item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

AS/400
C/400
DB2
IBM
MQ
MQSeries
Operating System/400
OS/400
RPG/400

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and 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.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 199.

- *MQSeries, Backup and Recovery*, SG24-5222 (available 3Q98)
- *Speak the Right Language with Your AS/400 System*, SG24-2154
- *AS/400 Communications Examples III*, GG24-4386
- *MQSeries for Windows Version 2.1 in a Mobile Environment*, SG24-2103
- *Internet Application Development with MQSeries and Java*, SG24-4896
- *Examples of Using MQSeries on WWW*, SG24-4882
- *An Early Look at Application Considerations Involved with MQSeries*, GG24-4469

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

C.3 Other Publications

These publications are also relevant as further information sources:

- *MQSeries Clients*, GC33-1632
- *MQSeries Intercommunication*, SC33-1872
- *MQSeries for Windows NT V5.0 Quick Beginning*, GC33-1871

© Copyright IBM Corp. 1998

- *MQSeries for AS/400 Version 4 Release 2 Administration Guide*, GC33-1956
- *MQSeries for AS/400 Version 4 Release 2 Application Programming Reference (RPG)*, SC33-1957

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com/.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/

- **PUBORDER** — to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.html). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

© Copyright IBM Corp. 1998

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

AIX	Advanced Interactive Executive (IBM′s flavor of UNIX)
API	Application Programming Interface
AS/400	Application System/400 (IBM)
CCSID	coded character set identifier
CL	command list
CPI-C	Common Programming Interface - Communications
CSI	CPI-C side information
DDM	distributed data management
DTAQ	data queue (AS/400)
EPM	enhanced editor for PM (OS/2)
GUI	Graphical User Interface
IBM	International Business Machines Corporation
ICF	Intercommunications Facility
IFS	AS/400 Internal File System
ILE	integrated language environment (IBM OS/400)
ITSO	International Technical Support Organization
MCA	Message Channel Agent
MQ	Message Queuing
MQI	Message Queuing Interface
MQM	MQ queue manager
MSGQ	message queue (AS/400)
OS/2	Operating System/2
OS/400	Operating System/400
PC	Personal Computer
PDM	Program Development Management
PSID	page set identifier
PTF	program temporary fix
QMQM	MQ queue manager (AS/400)
QSECOFR	MQ security officer
QSYS	AS/400 system library
RDB	Relational Database
SEU	Source Edit Utility
TCP/IP	Transmission Control Protocol / Internet Protocol
URL	Uniform Resource Locator
WWW	World Wide Web
XEDIT	Extended Editor (the IBM VM system editor that allows text input and revision)

© Copyright IBM Corp. 1998

Index

Numerics

A

B

C

© Copyright IBM Corp. 1998

D

E

F

G

H

I

N

O

P

Q

R

S

T

U

V

W

X

ITSO Redbook Evaluation

Using MQSeries on the AS/400
SG24-5236-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

© Copyright IBM Corp. 1998

SG24-5236-00
Printed in the U.S.A.