MQSeries® for OS/390®

Problem Determination Guide

Version 2 Release 1

GC34-5376-00

MQSeries® for OS/390®

Problem Determination Guide

Version 2 Release 1

GC34-5376-00

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix E, “Notices” on page 123.

First edition (January 1999)

This edition applies to MQSeries for OS/390 Version 2 Release 1 and to any subsequent releases and modifications until otherwise indicated in new editions.

This book is based on the *Problem Determination Guide* for MQSeries for MVS/ESA™ 1.2, GC33-0808-05.

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

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

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

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

Contents

Figures

Tables

Tables

About this book

This book is intended to help you determine the causes of MQSeries for OS/390 problems.

MQSeries for OS/390 is an OS/390 *queue manager* containing application programming services that allow you to code indirect program-to-program communication using *message queues*.

This book contains guidance about resolving MQSeries for OS/390 problems, dealing with the IBM support center, and handling APARs.

Who this book is for

This book is for those who are responsible for solving problems with MQSeries for OS/390 systems and application programs.

To use this book, you should be familiar with system programming concepts, OS/390 diagnostic procedures, and the structure and function of the MQSeries for OS/390 subsystems at your site. You should also be familiar with the other systems used with MQSeries for OS/390 at your site, for example, CICS® and IMS™.

Note: In this book, the term CICS is used to mean CICS Transaction Server for OS/390 and CICS for MVS/ESA, and the term IMS to mean IMS/ESA®, unless otherwise stated.

How to use this book

Use this book when you need assistance in resolving problems associated with your use of MQSeries for OS/390.

This book will help you decide whether the problem is caused by the way your system is set up, an application programming error, or a fault with the MQSeries for OS/390 program itself. The book guides you through problem analysis techniques.

Sometimes, you might mistake a failure within another environment, such as CICS or TSO, for a failure in MQSeries for OS/390. Wherever possible, this book helps you isolate MQSeries for OS/390 problems, and refers you to the appropriate publications if the error appears to be in another product.

If you decide that the problem is with the MQSeries for OS/390 code itself, this book tells you how to develop a set of keywords to use as a search argument in the IBM software support database, and how to communicate effectively with the IBM support center.

MQSeries publications

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

MQSeries cross-platform publications

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

- MQSeries for AIX V5.1
- MQSeries for AS/400 V4R2M1
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Digital OpenVMS V2.2
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for Sun Solaris V5.1
- MQSeries for Tandem NonStop Kernel V2.2
- MQSeries for VSE/ESA V2.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT V5.1

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

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

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

MQSeries Planning Guide
The *MQSeries Planning Guide*, GC33-1349, describes some key MQSeries concepts, identifies items that need to be considered before MQSeries is installed, including storage requirements, backup and recovery, security, and migration from earlier releases, and specifies hardware and software requirements for every MQSeries platform.

MQSeries Intercommunication
The *MQSeries Intercommunication* book, SC33-1872, defines the concepts of distributed queuing and explains how to set up a distributed queuing network in a variety of MQSeries environments. In particular, it demonstrates how to (1) configure communications to and from a representative sample of MQSeries products, (2) create required MQSeries objects, and (3) create and configure MQSeries channels. The use of channel exits is also described.

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

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

This book applies to the following MQSeries products only:

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

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

MQSeries Programmable System Management
The *MQSeries Programmable System Management* book, SC33-1482, provides both reference and guidance information for users of MQSeries events, Programmable Command Format (PCF) messages, and installable services.

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

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

This book is available in softcopy only.

MQSeries Application Programming Guide
The *MQSeries Application Programming Guide*, SC33-0807, provides guidance information for users of the message queue interface (MQI). It describes how to design, write, and build an MQSeries application. It also includes full descriptions of the sample programs supplied with MQSeries.

MQSeries Application Programming Reference
The *MQSeries Application Programming Reference*, SC33-1673, provides comprehensive reference information for users of the MQI. It includes: data-type descriptions; MQI call syntax; attributes of MQSeries objects; return codes; constants; and code-page conversion tables.

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

MQSeries Using C++
MQSeries Using C++, SC33-1877, provides both guidance and reference information for users of the MQSeries C++ programming-language binding to the MQI. MQSeries C++ is supported by these MQSeries products:

- MQSeries for AIX V5.1
- MQSeries for AS/400 V4R2M1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

MQSeries C++ is also supported by MQSeries clients supplied with these products and installed in the following environments:

- AIX®
- HP-UX
- OS/2®
- Sun Solaris
- Windows NT®
- Windows® 3.1
- Windows 95 and Windows 98

MQSeries Using Java®
MQSeries Using Java, SC34-5456, provides both guidance and reference information for users of the MQSeries Bindings for Java and the MQSeries Client for Java. MQSeries Java is supported by these MQSeries products:

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

MQSeries Administration Interface Programming Guide and Reference
The *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390, provides information for users of the MQAI. The MQAI is a programming interface that simplifies the way in which applications manipulate Programmable Command Format (PCF) messages and their associated data structures.

This book applies to the following MQSeries products only:

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

MQSeries Queue Manager Clusters
MQSeries Queue Manager Clusters, SC34-5349, describes MQSeries clustering. It explains the concepts and terminology and shows how you can benefit by taking advantage of clustering. It details changes to the MQI, and summarizes the syntax of new and changed MQSeries commands. It shows a number of examples of tasks you can perform to set up and maintain clusters of queue managers.

This book applies to the following MQSeries products only:

MQSeries for AIX V5.1

MQSeries for HP-UX V5.1
MQSeries for OS/2 Warp V5.1
MQSeries for OS/390 V2.1
MQSeries for Sun Solaris V5.1
MQSeries for Windows NT V5.1

MQSeries platform-specific publications

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

MQSeries for AIX

MQSeries for AIX Version 5 Release 1 Quick Beginnings, GC33-1867

MQSeries for AS/400®

MQSeries for AS/400 Version 4 Release 2.1 Administration Guide, GC33-1956

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

MQSeries for AT&T GIS UNIX®

MQSeries for AT&T GIS UNIX Version 2 Release 2 System Management Guide, SC33-1642

MQSeries for Digital OpenVMS

MQSeries for Digital OpenVMS Version 2 Release 2 System Management Guide, GC33-1791

MQSeries for Digital UNIX

MQSeries for Digital UNIX Version 2 Release 2.1 System Management Guide, GC34-5483

MQSeries for HP-UX

MQSeries for HP-UX Version 5 Release 1 Quick Beginnings, GC33-1869

MQSeries for OS/2 Warp

MQSeries for OS/2 Warp Version 5 Release 1 Quick Beginnings, GC33-1868

MQSeries for OS/390

MQSeries for OS/390 Version 2 Release 1 Licensed Program Specifications, GC34-5377

MQSeries for OS/390 Version 2 Release 1 Program Directory

MQSeries for OS/390 Version 2 Release 1 System Management Guide, SC34-5374

MQSeries for OS/390 Version 2 Release 1 Messages and Codes, GC34-5375

MQSeries for OS/390 Version 2 Release 1 Problem Determination Guide, GC34-5376

MQSeries link for R/3

MQSeries link for R/3 Version 1 Release 2 User's Guide, GC33-1934

MQSeries for SINIX and DC/OSx

MQSeries for SINIX and DC/OSx Version 2 Release 2 System Management Guide, GC33-1768

MQSeries for Sun Solaris

MQSeries for Sun Solaris Version 5 Release 1 Quick Beginnings, GC33-1870

MQSeries for Tandem NonStop Kernel

MQSeries for Tandem NonStop Kernel Version 2 Release 2 System Management Guide, GC33-1893

MQSeries for VSE/ESA™

MQSeries for VSE/ESA Version 2 Release 1 Licensed Program Specifications, GC34-5365

MQSeries for VSE/ESA Version 2 Release 1 System Management Guide, GC34-5364

MQSeries for Windows

MQSeries for Windows Version 2 Release 0 User's Guide, GC33-1822
MQSeries for Windows Version 2 Release 1 User's Guide, GC33-1965

MQSeries for Windows NT

MQSeries for Windows NT Version 5 Release 1 Quick Beginnings, GC34-5389
MQSeries for Windows NT Using the Component Object Model Interface, SC34-5387
MQSeries LotusScript Extension, SC34-5404

MQSeries Level 1 product publications

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

MQSeries: Concepts and Architecture, GC33-1141

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

MQSeries for UnixWare Version 1 Release 4.1 User's Guide, SC33-1379

Softcopy books

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

BookManager® format

The MQSeries library is supplied in IBM BookManager format on a variety of online library collection kits, including the *Transaction Processing and Data* collection kit, SK2T-0730. You can view the softcopy books in IBM BookManager format using the following IBM licensed programs:

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

HTML format

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

- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1 (compiled HTML)
- MQSeries link for R/3 V1.2

The MQSeries books are also available in HTML format from the MQSeries product family Web site at:

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

Portable Document Format (PDF)

PDF files can be viewed and printed using the Adobe Acrobat Reader.

If you need to obtain the Adobe Acrobat Reader, or would like up-to-date information about the platforms on which the Acrobat Reader is supported, visit the Adobe Systems Inc. Web site at:

```
http://www.adobe.com/
```

PDF versions of relevant MQSeries books are supplied with these MQSeries products:

- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1
- MQSeries link for R/3 V1.2

PDF versions of all current MQSeries books are also available from the MQSeries product family Web site at:

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

PostScript format

The MQSeries library is provided in PostScript (.PS) format with many MQSeries Version 2 products. Books in PostScript format can be printed on a PostScript printer or viewed with a suitable viewer.

Windows Help format

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

MQSeries information available on the Internet

MQSeries Web site

The MQSeries product family Web site is at:

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

By following links from this Web site you can:

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

Related publications

You might find the books listed below helpful in determining the causes of problems with MQSeries for OS/390.

CICS Transaction Server for OS/390

- *CICS Intercommunication Guide*, SC33-1695
- *CICS Performance Guide*, SC33-1699
- *CICS Customization Guide*, SC33-1683
- *CICS Messages and Codes*, GC33-1694
- *CICS Problem Determination Guide*, GC33-1696

CICS for MVS/ESA Version 4

- *CICS Intercommunication Guide*, SC33-1181
- *CICS Performance Guide*, SC33-1183
- *CICS Customization Guide*, SC33-1165
- *CICS Messages and Codes*, GC33-1177
- *CICS Problem Determination Guide*, SC33-1176

IMS/ESA

- *IMS/ESA Messages and Codes*, SC26-8028
- *IMS/ESA Diagnosis Guide and Reference*, SC26-8848

OS/390

- *MVS System Commands*, GC28-1781
- *MVS Diagnosis: Tools and Service Aids*, LY28-1085
- *MVS IPCS Commands*, GC28-1756
- *MVS Initialization and Tuning Reference*, SC28-1752
- *MVS System Messages* (Volumes 1 through 5), GC28-1784 through GC28-1788
- *MVS System Codes*, GC28-1780

Other products

- *OS/390 C/C++ Programming Guide*, SC09-2362
- *Language Environment Debugging Guide and Run-Time Messages*, SC28-1942
- *VTAM Messages*, GC31-8368
- *TCP/IP for MVS Messages and Codes*, SC31-7132
- *OS/390 Security Server (RACF) Messages and Codes*, SC28-1918

What is new for this version

- The minimum levels for many of the items of software required to use MQSeries for OS/390 have changed. These are described in the *MQSeries Planning Guide*.
- There are several changes to the installation process, and some new libraries. These are described in the *MQSeries for OS/390 Program Directory*.
- The OS/390 Automatic Restart Manager (ARM) is now supported.
- MQSeries now supports clustering. This is described in the *MQSeries Queue Manager Clusters* manual.
- MQSeries for OS/390 now supports C++. This is described in the *MQSeries Using C++* manual.
- MQSeries Batch and TSO applications can now participate in 2-phase commit protocols with other RRS-enabled products, coordinated by OS/390 Resource Recovery Services (RRS).
- If you are using TCP/IP for distributed queuing, you can now use the OpenEdition® sockets interface.
- The CICS bridge is now incorporated into MQSeries for OS/390.
- MQSeries can now be used in conjunction with the OS/390 Workload Manager.
- The operations and control panels have been extended and reorganized to accommodate the new clustering function. The panels can now also be used for the security commands. A number of actions have been consolidated into a new PERFORM action.

For more information about these features, see the manuals mentioned or the *MQSeries for OS/390 System Management Guide*.

Part 1. Approaching problem determination

Chapter 1. Introduction to problem determination

This book tells you how to find the reasons for problems with MQSeries for OS/390. This process is called problem determination. You usually start with a symptom, or set of symptoms, and trace them back to their cause.

You should not confuse problem determination with problem solving; however, the process of problem determination often enables you to solve a problem. For example:

- If you find that the cause of the problem is an error in an application program, you can solve the problem by fixing the error.
- If you find that the cause of the problem is an error in your system programming (such as an error in resource definition), you can solve the problem by correcting the error.

However, you may not always be able to solve a problem after determining its cause. For example:

- A performance problem may be caused by a limitation of your hardware.
- You may find that the cause of the problem is in MQSeries for OS/390 itself. If this happens, you need to contact your IBM support center for a solution.

How this book can help you

The approach in this book is to start with the symptoms of the problem, and try to use these symptoms to classify it. For each class of problem, possible causes are suggested, together with techniques you can use to establish what the cause is.

It is always assumed that the problem has a simple cause to start with (for example, an application programming error). If, as a result of investigation, it becomes clear that the cause of the problem is not straightforward, then you must consider possible causes that may be more difficult to determine (and solve). Having exhausted all other possibilities, consider the possibility that the cause of the problem may be in MQSeries itself, in which case you would need to get help from IBM.

How this book is organized

This book contains the following sections:

Part 1, “Approaching problem determination”
This section outlines the simple preliminary checks and tests that you should perform to try and establish the cause of the problem. You should find that working through this section is all that is required to solve your problem.

Part 2, “Dealing with the problem”
This section tells you how to deal with more complex MQSeries problems, diagnosed in the previous section. It covers:

- Program abends
- Waits and loops
- Performance problems
- Incorrect output

Part 3, "Diagnostic aids and techniques"
This section covers the tools you can use in solving your MQSeries problems. It discusses the use of dumps and traces, and gives a list of other possible sources of information about the problem.

Part 4, "Searching the database for solutions to similar problems"
This section tells you how to develop a set of keywords to search the IBM software support database for solutions to similar problems.

Part 5, "Working with IBM to solve your problem"
This section tells you what to expect if you need to contact the IBM support center for help with your problem. It also covers the APAR process.

Preliminary checks

Before you start problem determination in detail, it is worth considering the facts to see if there is an obvious cause of the problem, or a promising area in which to start your investigation. This approach to debugging can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

The cause of your problem could be in any of the following:

- Queue manager address space
- Channel initiator address space
- CICS address space
- IMS region
- Batch or TSO address space
- The OS/390 system (including APPC, ARM, RRS, or TCP/IP)
- The network
- Another system, for example a queue manager on another platform or an MQSeries client

The sections that follow raise some fundamental questions that you will need to consider.

As you go through the questions, make a note of anything that might be relevant to the problem. Even if your observations do not suggest a cause straightaway, they could be useful later if you have to carry out a systematic problem determination exercise.

Has MQSeries for OS/390 run successfully before?

If MQSeries has not run successfully on OS/390 before, it is likely that you have not yet set it up correctly. You will need to see the information about installing and customizing the queue manager in the *MQSeries for OS/390 System Management Guide* for guidance on doing this.

Check that message CSQ9022I was issued in response to the START QMGR command (indicating normal completion). You should also check that the installation verification program (IVP) was run successfully.

Are there any error messages or return codes explaining the problem?

MQSeries for OS/390 error messages have the prefix CSQ; if you receive any messages with this prefix (for example, in the console log, or the CICS log), look in the *MQSeries for OS/390 Messages and Codes* manual for an explanation. For messages with a different prefix, look in the appropriate messages and codes manual for a suggested course of action that might help you resolve the problem.

Any unusual messages associated with the startup of MQSeries for OS/390, or issued while the system was running before the error occurred, might indicate some system problem that prevented your application from running successfully.

If your application gets a return code indicating that an MQI call has failed, see the *MQSeries Application Programming Reference* manual for a description of that return code.

Can the problem be reproduced?

If the problem is reproducible, consider the conditions under which it can be reproduced:

- Is it caused by a command? If so, is the command issued from the OS/390 console, from CSQUTIL, from a program written to put things onto the SYSTEM.COMMAND.INPUT queue, or on the operations and control panels?

  Does the command work if it is entered by another method? If the command works when it is entered at the console, but not otherwise, check that the command server has not stopped, and that the queue definition of the SYSTEM.COMMAND.INPUT queue has not been changed.
- Is it caused by an application? If so, does it fail in CICS, IMS, TSO, or batch? Does it fail on all MQSeries systems, or only on some?
- Can you identify any application that always seems to be running in the system when the problem occurs? If so, examine the application to see if it is in error.

Have any changes been made since the last successful run?

When you are considering changes that might recently have been made, think about MQSeries, and also about the other programs it interfaces with, the hardware, and any new applications. Consider also the possibility that a new application that you don't yet know about might have been run on the system.

- Has your initialization procedure been changed?

  Consider whether that might be the cause of the problem. Have you changed any data sets, or changed a library definition? In addition, check for error messages sent to the console during initialization.
- Have you changed any queue definitions or security profiles?

  Consider whether some of your queues have been altered so that they are members of a cluster. This might mean that messages arrive from different sources (for example, other queue managers or applications).
- Has any of the software on your OS/390 system been upgraded to a later release?

  Changes to levels of co-requisite software like OS/390 or LE might mean that you have to also make some changes to MQSeries.

- Do your applications deal with return codes that they might get as a result of any changes you have made?

Has the application run successfully before?

If the problem appears to involve one particular application, consider whether the application has run successfully before.

Before you answer **Yes** to this question, consider the following:

- Have any changes been made to the application since it last ran successfully?

  If so, it is likely that the error lies somewhere in the new or modified part of the application. Take a look at the changes and see if you can find an obvious reason for the problem.

- Have all the functions of the application been fully exercised before?

  Could it be that the problem occurred when part of the application that had never been invoked before was used for the first time? If so, it is likely that the error lies in that part of the application. Try to find out what the application was doing when it failed, and check the source code in that part of the program for errors.

  If a program has been run successfully on many previous occasions, check the current queue status and files that were being processed when the error occurred. It is possible that they contain some unusual data value that causes a rarely used path in the program to be invoked.

- Does the application check all return codes?

  Could it be that your system has been changed, perhaps in a minor way, but your application does not check the return codes it receives as a result of the change. For example:

  – Does your application assume that the queues it accesses can be shared? If a queue has been redefined as exclusive, can your application deal with return codes indicating that it can no longer access that queue?

  – Have any security profiles been altered? An **MQOPEN** call could fail because of a security violation; can your application recover from the resulting return code?

- Does the application expect particular message formats?

  If a message with an unexpected message format has been put onto a queue (for example, a message from a queue manager on a different platform) it might require data conversion or another different form of processing.

- Does the application run on other MQSeries for OS/390 systems?

  Could it be that there is something different about the way that this queue manager is set up which is causing the problem? For example, have the queues been defined with the same maximum message length, or priority?

The application has not run successfully before

If your application has not yet run successfully, you need to examine it carefully to see if you can find any errors.

Before you look at the code, examine the output from the **translator**, the **compiler** or **assembler**, and the **linkage editor**, to see if any errors have been reported. If your application fails to translate, compile/assemble, or link-edit into the load library, it will also fail to run if you attempt to invoke it. See the *MQSeries Application Programming Guide* for information about building your application, and for examples of the job control language (JCL) statements required.

For batch and TSO programs, check that the correct stub has been included. There is one batch stub and two RRS stubs. If you are using RRS, check that you are not using the **MQCMIT** and **MQBACK** calls with the CSQBRSTB stub; use the CSQBRRSI stub if you want to continue using these calls with RRS.

For CICS programs, check that the program, the MQSeries stub, and the CICS stub have been linked in the correct order. Also check that your program or transaction is defined to CICS.

For IMS programs, check that the link includes the program, the MQSeries stub, and the IMS language interface module, and that the correct entry point has been specified. Note that a program that is loaded dynamically from an IMS program must have the stub and language interface module linked also if it is to use MQSeries.

If the documentation shows that each of these steps was accomplished without error, you should consider the coding logic of the application. Do the symptoms of the problem indicate the function that is failing and, therefore, the piece of code in error? See “Common programming errors” on page 9 for some examples of common errors that cause problems with MQSeries applications.

Does the problem affect specific parts of the network?

You might be able to identify specific parts of the network that are affected by the problem (remote queues, for example). If the link to a remote queue manager is not working, the messages cannot flow to a target queue on the target queue manager. Check that the connection between the two systems is available, and that the channel initiator has been started.

Check that messages are reaching the transmission queue, and check the local queue definition of the transmission queue, and any remote queues.

If you are using MQSeries clusters, check that the clustering definitions have been set up correctly.

Have you made any network-related changes that might account for the problem?

Have you changed any MQSeries definitions, or any CICS or IMS definitions? Check the triggering attributes of the transmission queue, and look in the CICS log, if possible, to see if there has been a problem with the CICS transactions on the receiving system (if you are using CICS ISC for distributed queuing).

Does the problem occur at specific times of the day?

If the problem occurs at specific times of day, it could be that it is dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, and so these are the times when load-dependent problems are most likely to occur. (If your network extends across more than one time zone, peak system loading might seem to occur at some other time of day.)

If you think that your MQSeries for OS/390 system has a performance problem, refer to Chapter 5, “Dealing with performance problems” on page 29.

Does the problem affect all users of the application?

If the problem only affects some users, is this because some users do not have the correct security authorization? See the *MQSeries for OS/390 System Management Guide* for information about all aspects of security for MQSeries for OS/390.

Does the problem occur with all OS/390, CICS, or IMS systems?

If the problem only occurs when you access a particular OS/390, IMS, or CICS system, consider what is different about this system. Also consider whether any changes have been made to the system that might affect the way it interacts with MQSeries.

Is the problem intermittent?

An intermittent problem could be caused by failing to take into account the fact that processes can run independently of each other. For example, a program might issue an **MQGET** call, without specifying WAIT, before an earlier process has completed. You might also encounter this if your application tries to get a message from a queue while it is in syncpoint (that is, before it has been committed or backed out).

Have you applied any APARs or PTFs?

If an APAR or PTF has been applied to MQSeries for OS/390, check that no error message was produced. If the installation was successful, check with the IBM support center for any APAR or PTF error.

If an APAR or PTF has been applied to any other program, consider the effect it might have on the way MQSeries interfaces with it.

Ensure that you have followed any instructions in the APAR which affect your system. (For example, you might have to redefine a resource.)

Common programming errors

The examples that follow illustrate the most common causes of problems encountered while running MQSeries programs. You should consider the possibility that the problem with your system could be caused by one of these errors.

- Making incorrect assumptions about the attributes of a queue. This could include assuming that queues can be shared when they are exclusive, and assuming that queues are not part of a cluster when they are.
- Trying to access queues and data without the correct security authorization.
- Linking a program with no stub, or with the wrong stub (for example, a TSO program with the CICS stub). This can cause an `X'0C4'` or other abend.
- Passing incorrect or invalid parameters in an MQI call; if the wrong number of parameters are passed, no attempt can be made to complete the completion code and reason code fields, and the task is abended. (This is an `X'0C4'` abend.)

  This problem might occur if you attempt to run an application on an earlier version of MQSeries than it was written for, where some of the MQI values are invalid.
- Failing to define the MQSeries modules to OS/390 correctly (this causes an `X'0C4'` abend in CSQYASCP).
- Failing to check return codes from MQI requests.

  This problem might occur if you attempt to run an application on a later version of MQSeries than it was written for, where new return codes have been introduced that are not checked for.
- Failing to open objects with the correct options needed for later MQI calls, for example using the **MQOPEN** call to open a queue but not specifying the correct options to enable the queue for subsequent **MQGET** calls.
- Failing to initialize *MsgId* and *CorrelId* correctly.
- Using incorrect addresses.
- Using storage before it has been initialized.
- Passing variables with incorrect lengths specified.
- Passing parameters in the wrong order.

What to do next

Perhaps the preliminary checks have enabled you to find the cause of the problem. If so, you should now be able to resolve it, possibly with the help of other books in the MQSeries library (see “Related publications” on page xvii) and in the libraries of other licensed programs.

If you have not yet found the cause, you must start to look at the problem in greater detail. Begin by running the tests described in Chapter 2, “Identifying the cause of the problem” on page 11 to try to determine what type of problem you have.

Chapter 2. Identifying the cause of the problem

The purpose of this chapter is to help you identify the cause of your problem if the preliminary checks have not enabled you to solve it.

Before you read this chapter, you should have worked through "Preliminary checks" on page 4. You will also need access to the console log, and to diagnostic tools.

When you have established that no changes have been made to your system, and that there are no problems with your application programs, choose the option that best describes the symptoms of your problem.

- "Have you obtained some incorrect output?"
- "Have you received an unexpected error message?" on page 12
- "Has there been an abend?" on page 12
- "Have you failed to receive a response from an MQSeries command?" on page 13
- "Is there a problem with the MQSeries queues?" on page 14
- "Is your application or MQSeries for OS/390 running slowly?" on page 16
- "Has your application or MQSeries for OS/390 stopped processing work?" on page 17

If none of these symptoms describe your problem, consider whether it might have been caused by another component of your system.

Have you obtained some incorrect output?

"Incorrect output" might be regarded as any sort of output that you were not expecting. However, use this term with care in the context of problem determination because it might be a secondary effect of some other type of error. For example, looping could be occurring if you get any sort of repetitive output, even though that output is not what you expected. Also, MQSeries responds to many errors it detects by sending error messages. You might regard these messages as "incorrect output", but they are only symptoms of another type of problem. If you have received an error message from MQSeries that you were not expecting, refer to "Have you received an unexpected error message?" on page 12.

If your application has not received a message that it was expecting, has received a message containing unexpected or corrupted information, or has received a message that it was not expecting (for example, one that was destined for a different application), refer to Chapter 6, "Dealing with incorrect output" on page 35.

Have you received an unexpected error message?

MQSeries for OS/390 error messages are prefixed with the letters CSQ.

If you get an unexpected MQSeries error message (for example, in the console log, or the CICS log), look in the *MQSeries for OS/390 Messages and Codes* manual for an explanation. If you get an error message from another IBM program, or from the operating system, look in the messages and codes manual from the appropriate library for an explanation of what it means.

The *MQSeries for OS/390 Messages and Codes* manual might give you enough information to resolve the problem quickly, or it might redirect you to another manual for further guidance. If you are unable to deal with the message, you might have to contact the IBM support center for help.

If your application has received an unexpected return code from MQSeries, see the *MQSeries Application Programming Reference* manual for information about how your application should deal with MQSeries return codes.

Has there been an abend?

If your application has stopped running, this could be caused by an abnormal termination (abend).

You are notified of an abend in one of the following places, depending on what type of application you are using:

Batch Your listing shows the abend.

CICS You see a CICS transaction abend message. If your task is a terminal task, this message appears on your screen. If your task is not attached to a terminal, the message appears on the CICS CSMT log.

IMS In all cases, you see a message at the IMS master terminal and in the listing of the dependent region involved. If an IMS transaction that had been entered from a terminal was being processed, an error message is also sent to that terminal.

TSO You might see a TSO message with a return code on your screen. (This depends on the way your system is set up, and the type of error.)

Abends can be caused by the user ending the task being performed before it terminates normally (for example, purging a CICS transaction), or by an error in an application program.

For some abends, an address space dump is produced. For CICS transactions, a transaction dump showing the storage areas of interest to the transaction is provided.

- If an application passes some data, the address of which is no longer valid, a dump is sometimes produced in the user's address space.

  Note: For a batch dump, the dump is formatted and written to SYSUDUMP. For information about SYSUDUMPs, refer to "SYSUDUMP information" on page 64. For CICS, a system dump is written to the SYS1DUMP data sets, as well as a transaction dump being taken.

- If a problem with MQSeries for OS/390 itself causes an abend, an abend code of `X'5C6'` or `X'6C6'` is returned, along with an abend reason code. This uniquely describes the cause of the problem. Refer to “MQSeries for OS/390 abends” on page 44 for information about the abend codes, and see the *MQSeries for OS/390 Messages and Codes* manual for an explanation of the reason code.

If your program has terminated abnormally, refer to Chapter 3, “Dealing with program abends” on page 21. If your system has terminated abnormally, and you want to analyze the dump produced, refer to Chapter 8, “MQSeries dumps” on page 51. This chapter tells you how to format the dump, and how to interpret the data contained in it.

Have you failed to receive a response from an MQSeries command?

If you have issued an MQSeries command, but you have not received a response, consider the following questions:

- Is the queue manager still running, or did your command cause an abend?

  Look for error messages indicating an abend, and if one occurred, refer to Chapter 8, “MQSeries dumps” on page 51.

- Is the command server running?

  Use the DISPLAY CMDSERV command at the OS/390 console to check the status of the command server.

  – If you do not receive a response to this command, refer to “Has your application or MQSeries for OS/390 stopped processing work?” on page 17.

  – If the response to this command indicates that the command server is not running, use the START CMDSERV command to start it.

- Has a reply been sent to the dead-letter queue?

  Use the DISPLAY QMGR DEADQ command to find out the name of the system dead-letter queue (if you do not know what it is).

  Use this name in the DISPLAY QUEUE command with the CURDEPTH attribute to see if there are any messages on the queue.

  The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code describing the problem. (See the *MQSeries Application Programming Reference* manual for information about the dead-letter header structure.)

- If the command was issued from the operations and control panels or the utility program, were any error messages issued indicating the nature of the error?

- Are the correct queues defined, (the system-command input queue, the system-command reply model queue, and the reply-to queue), and were the **MQOPEN** calls successful? If you are using the system-command reply model queue, check that it was defined correctly. In particular, ensure that it has a DEFTYPE of PERMDYN.

  If you are using clusters, you need to define the SYSTEM.CLUSTER.COMMAND.QUEUE to use commands relating to cluster processing.

- Are the queues enabled for PUTs and GETs?

  Use the DISPLAY QUEUE command from the console to check, for example, DISPLAY QUEUE(SYSTEM.COMMAND.INPUT) PUT GET.

- Is the *WaitInterval* parameter set to a sufficiently long time?

  If your **MQGET** call has timed out, you will see a completion code of 2 and a reason code of 2033 (MQRC_NO_MSG_AVAILABLE). (See the *MQSeries Application Programming Guide* for information about the *WaitInterval* parameter, and completion and reason codes from **MQGET**.)

- If you are using your own application program to put commands onto the system-command input queue, do you need to take a syncpoint?

  Unless you have specifically excluded your request message from syncpoint, you will need to take a syncpoint before attempting to receive reply messages.

- Are the *MaxDepth* and *MaxMsgL* parameters of your queues set sufficiently high?

  See the *MQSeries for OS/390 System Management Guide* for information about defining the system-command input queue and the reply-to queue.

- Are you using the *CorrelId* and *MsgId* parameters correctly?

  Use the DISPLAY QUEUE command from the console, for example, DISPLAY QUEUE(SYSTEM.COMMAND.REPLY.MODEL) CURDEPTH, to see if there are messages on the reply-to queue that you have not received.

  Set the values of *MsgId* and *CorrelId* in your application to ensure that you receive all messages from the queue. See the *MQSeries Application Programming Guide* for information about using these fields.

See the *MQSeries for OS/390 System Management Guide* for information about the different methods you can use to enter MQSeries commands.

Is there a problem with the MQSeries queues?

If you suspect that there is a problem affecting the queues on your subsystem, use the operations and control panels to issue the DISPLAY QUEUE(SYSTEM.COMMAND.INPUT) command, in order to display the system-command input queue.

- If the system responds to the command, at least one queue is working, so follow the procedure shown in “Are some of your queues working?” on page 15.
- If the system does not respond to the command, check that the command server is running, as follows:
  1. Use the DISPLAY CMDSERV command at the OS/390 console to display the status of the command server.
  2. If the command server is not running, start it using the START CMDSERV command.
  3. If the command server is running, issue the DISPLAY QUEUE command, using the name of the system-command input queue and the CURDEPTH and MAXDEPTH attributes to define the data displayed.

     If these values show that the queue is full, and the command server has been started, this indicates that messages are not being read from the queue.

4. Try stopping the command server and then restarting it, responding to any error messages that are produced.
5. Try to issue the display command from the operations and control panels again to see if it is working now.

- If the system still does not respond, then the problem could be with the whole subsystem. Try stopping and restarting the queue manager, responding to any error messages that are produced.

  If the problem still occurs after restart, contact your IBM support center for help (see Chapter 12, “IBM program support” on page 97).

Are some of your queues working?

If you suspect that the problem occurs with only a subset of queues, select the name of a local queue that you think is having problems.

1. Use the operations and control panels or the console to display information about the queue.
2. Use the data displayed to do the following checks:
   - If CURDEPTH is at MAXDEPTH, this might indicate that the queue is not being processed. Check that all applications that use the queue are running normally (for example, check that transactions in your CICS system are running or that applications started in response to Queue Depth High events are running).
   - If CURDEPTH is not at MAXDEPTH, check the following queue attributes to ensure that they are correct:
     – If triggering is being used:
       - Is the trigger monitor running?
       - Is the trigger depth too big?
       - Is the process name correct?
       - Have **all** the trigger conditions been met?

     – Can the queue be shared? If not, another application (batch, IMS, or CICS) might already have it open for input.

     – Is the queue enabled appropriately for GET and PUT?
   - Check the OPPROCS and IPPROCS to see how many tasks are putting messages on to, and getting messages from the queue. If there are no application processes getting messages from the queue, determine why this is (for example, because the applications need to be started, a connection has been disrupted, or because the **MQOPEN** call has failed for some reason).
   - Check to see if the queue is part of a cluster (from the CLUSTER or CLUSNL attribute). If it is, verify that the queue manager that hosts the queue is still active in the cluster.

If you are unable to solve the problem, contact your IBM support center for help (see Chapter 12, “IBM program support” on page 97).

Does the problem affect only remote or cluster queues?

If the problem affects only remote or cluster queues, check the following:

- Check that the programs which should be putting messages to the remote queues have run successfully (see Chapter 6, “Dealing with incorrect output” on page 35).
- Use APPC or TCP/IP commands as appropriate to check whether the link between the two systems is active.
- If you use triggering to start the distributed queuing process, check that the transmission queue has triggering set on and that the queue is get-enabled.
- If necessary, start the channel or the listener manually. See the *MQSeries Intercommunication* manual for information about how to do this.
- Check the channel status. See the *MQSeries Intercommunication* manual for information about how to do this.
- Check your process definitions and your channel definitions.
- If you are using CICS ISC for distributed queuing, check the following:
  1. Find out which CICS system you are using to move messages to the remote queue manager. Your system documentation should include this information.
  2. Use CEMT INQ TASK to check that the distributed queuing processes are running. If not, check the console log and the CICS CSMT log for error messages.
  3. Use the CKQC transaction to check that the CICS system has an active connection to its queue manager, and use CEMT INQ CONNECTION to check that the LU 6.2 link between the two systems is active.
  4. If you use triggering to start the distributed queuing process, use CKQC to check that the trigger monitor is running.

See the *MQSeries for OS/390 System Management Guide* for information about how to use distributed queuing, and the *MQSeries Intercommunication* manual for information about how to define channels.

Is your application or MQSeries for OS/390 running slowly?

If your application is running slowly, this could indicate that it is in a loop, or waiting for a resource that is not available.

This could also be caused by a performance problem. Perhaps it is because your system needs tuning, or because it is operating near the limits of its capacity. This type of problem is probably worst at peak system load times, typically at mid-morning and mid-afternoon. (If your network extends across more than one time zone, peak system load might seem to you to occur at some other time.)

If you find that performance degradation is not dependent on system loading, but happens sometimes when the system is lightly loaded, then a poorly designed application program is probably to blame. This could manifest itself as a problem that only occurs when certain queues are accessed.

The following symptoms might indicate that MQSeries for OS/390 is running slowly:

- If your system is slow to respond to commands.

- If repeated displays of the queue depth indicate that the queue is being processed slowly for an application with which you would expect a large amount of queue activity.

You can find guidance on dealing with waits and loops in Chapter 4, “Dealing with waits and loops” on page 23, and on dealing with performance problems in Chapter 5, “Dealing with performance problems” on page 29.

Has your application or MQSeries for OS/390 stopped processing work?

There are several reasons why your system would unexpectedly stop processing work. These include:

Application problems
An application programming error could mean that the program branches away from its normal processing, or the application could get in a loop. There could also have been an application abend.

MQSeries problems
Your queues could have become disabled for **MQPUT** or **MQGET** calls, the dead-letter queue could be full, or MQSeries for OS/390 could be in a wait state, or a loop.

OS/390 and other system problems
OS/390 could be in a wait state, or CICS or IMS could be in a wait state or a loop. There might have been a system abend.

Issue the DISPLAY THREAD(*) command to check if the MQSeries system is running (see the *MQSeries Command Reference* manual for information about the command). If the queue manager has stopped running, look for any message that might explain the situation. Messages appear on the OS/390 console, or on your terminal if you are using the operations and control panels. The OS/390 command

```
DISPLAY R,L
```

lists messages with outstanding replies. Check to see whether any of these are relevant. In some circumstances, for example, when it has used all its active logs, MQSeries for OS/390 waits for operator intervention.

Look for any messages saying that the MQSeries job has abnormally terminated. If you find one, it means that an MQSeries system abend has occurred and that MQSeries for OS/390 is no longer running. If you get a message for which the system action is to terminate MQSeries, find out if a system dump was produced, and turn to Chapter 8, “MQSeries dumps” on page 51.

If no error messages have been issued, perform the following procedure to determine what is causing the problem:

1. Issue the OS/390 command

   ```
   DISPLAY A,xxxxMSTR
   ```

   (where xxxx is the MQSeries for OS/390 subsystem name). If you receive a message telling you that the subsystem has not been found, this indicates that the subsystem has terminated. This could be caused by an abend or by operator shutdown of the system.

2. If the subsystem is running, you will receive message IEE105I. This message includes the *CT=nnnn* field which contains information about the processor time being used by the subsystem. Note the value of this field, and reissue the command.
   - If the *CT=* value has not changed, this indicates that the subsystem is not using any processor time. This could indicate that the subsystem is in a wait state (or that it has no work to do).
   - If the *CT=* value has changed dramatically, and continues to do so over repeated displays, this could indicate that the subsystem is in a loop.
   - If the reply indicates that the subsystem is now not found, this indicates that it was in the process of terminating when the first command was issued. If a dump is being taken, the subsystem might take a while to terminate. A message is produced at the console before terminating.

Note: You can also check that the channel initiator is running by issuing the OS/390 command `DISPLAY A,xxxxCHIN`.

Consider also that MQSeries for OS/390 might still be running, but only slowly. If it *is* running slowly, you probably have a performance problem. If so, read "Is your application or MQSeries for OS/390 running slowly?" on page 16 to confirm this before going on. Refer to Chapter 5, "Dealing with performance problems" on page 29 for advice about what to do next.

Where to look next

Table 1. Where to look next

Type of problem	**Where to look next**
Program abend	Chapter 3, "Dealing with program abends" on page 21
System abend	Chapter 8, "MQSeries dumps" on page 51
Wait	Chapter 4, "Dealing with waits and loops" on page 23
Loop	Chapter 4, "Dealing with waits and loops" on page 23
Performance problem	Chapter 5, "Dealing with performance problems" on page 29
Incorrect output	Chapter 6, "Dealing with incorrect output" on page 35
Unexpected message	*MQSeries for OS/390 Messages and Codes* manual
Application error	*MQSeries Application Programming Guide* and *MQSeries Application Programming Reference* manual

If you have decided that the problem is with the MQSeries code itself, and that you should refer it to the IBM support center, you can get advice on dealing with the support center in Chapter 12, "IBM program support" on page 97.

Part 2. Dealing with the problem

Chapter 3. Dealing with program abends

Program abends can be caused by applications failing to check, and respond to, reason codes from MQSeries. For example, if a message has not actually been received, using fields that would have been set up in the message for calculation could cause `X'0C4'` or `X'0C7'` abends (ASRA abends in CICS).

The following pieces of information indicate a program abend:

- Error messages from MQSeries in the console log
- CICS error messages
- CICS transaction dumps
- IMS region dumps
- IMS messages on user or master terminal
- Program dump information in batch or TSO output
- Abend messages in batch job output
- Abend messages on the TSO screen

If you have an abend code, see one of the following manuals for an explanation of the cause of the abend:

- For MQSeries for OS/390 abends (abend codes `X'5C6'` and `X'6C6'`), the *MQSeries for OS/390 Messages and Codes* manual
- For batch abends, the *MVS System Codes* manual
- For CICS abends, the *CICS Messages and Codes* manual
- For IMS abends, the *IMS/ESA Messages and Codes* manual

Batch abends cause an error message containing information about the contents of registers to appear in the syslog. TSO abends cause an error message containing similar information to be produced on the TSO screen. A SYSUDUMP is taken if there is a SYSUDUMP DD statement for the step (refer to Chapter 8, “MQSeries dumps” on page 51 for information about SYSUDUMPs).

CICS transaction abends are recorded in the CICS CSMT log, and a message is produced at the terminal (if there is one). A CICS AICA abend indicates a possible loop. Refer to “Dealing with loops” on page 26 for more information. If you have a CICS abend, using CEDF and the CICS trace might help you to find the cause of the problem. See the *CICS Problem Determination Guide* for more information.

IMS transaction abends are recorded on the IMS master terminal, and an error message is produced at the terminal (if there is one). If you have an IMS abend, see the *IMS/ESA Diagnosis Guide and Reference* manual.

Chapter 4. Dealing with waits and loops

To perform the tests shown in this chapter, you will need access to the OS/390 console, and to be able to issue operator commands.

Waits and loops are characterized by nonresponsiveness. However, it can be quite difficult to distinguish between waits, loops, and poor performance.

Any of the following symptoms could be caused by a wait or a loop, or by a badly tuned or overloaded system:

- An application that appears to have stopped running (if MQSeries for OS/390 is still responsive, this is probably an application problem)
- An MQSeries command that does not produce a response
- Excessive use of processor time (CPU)

Distinguishing between waits and loops

For the purpose of problem determination, a wait state is regarded as the state in which the execution of a task has been suspended. That is, the task has started to run, but has been suspended without completing, and has subsequently been unable to resume.

A problem identified as a wait in your system could be caused by any of the following:

- A wait on an MQI call
- A wait on a CICS or IMS call
- A wait for another resource (for example, file I/O)
- An ECB wait
- The CICS or IMS region waiting
- TSO waiting
- MQSeries for OS/390 waiting
- An apparent wait, caused by a loop

A loop is the repeated execution of some code. If you have not planned the loop, or if you have designed it into your application but it does not terminate for some reason, you get a set of symptoms that vary depending on what the code is doing, and how any interfacing components and products react to it. In some cases, at first, a loop might be diagnosed as a wait or performance problem, because the looping task competes for system resources with other tasks that are not involved in the loop.

An apparent loop problem in your system could be caused by any of the following:

- A loop in application logic
- A loop with MQI calls
- A loop with CICS or IMS calls
- A loop in CICS or IMS code
- A loop in MQSeries for OS/390

Symptoms of waits and loops

Any of the following symptoms could be caused by a wait, a loop, or by a badly tuned or overloaded system:

- Timeouts on **MQGET** WAITs
- Batch jobs suspended
- TSO session suspended
- CICS task suspended
- Queues becoming full, and not being processed
- System commands not accepted, or producing no response

Dealing with waits

When investigating what appears to be a problem with tasks or subsystems waiting, it is necessary to take into account the environment in which the task or subsystem is running.

It might be that your OS/390 system is generally under stress. In this case, there will be many symptoms. If there is not enough real storage, jobs will experience waits at paging interrupts or swap-outs. Input/output (I/O) contention or high channel path usage can also cause waits.

You can use standard monitoring tools, such as *Resource Monitoring Facility* (RMF™) to diagnose such problems. Normal OS/390 tuning techniques should be used to resolve them.

Is a batch or TSO program waiting?

Consider the following points:

- Your program might be waiting on another resource, for example, a VSAM control interval (CI) that another program is holding for update.
- Your program might be waiting for a message that has not yet arrived. This might be normal behavior if, for example, it is a server program that constantly monitors a queue. Alternatively, your program might be waiting for a message that has arrived, but has not yet been committed.

If you suspect that your program has issued an MQI call that did not involve an **MQGET** WAIT, and control has not returned from MQSeries, take an SVC dump of both the batch or TSO job, and the MQSeries for OS/390 subsystem before cancelling the batch or TSO program. Refer to “Is MQSeries for OS/390 waiting?” on page 25.

Also consider that the wait state might be the result of a problem with another program, such as an abnormal termination (see “Messages do not appear on the queue” on page 35), or in MQSeries for OS/390 itself (see “Is MQSeries for OS/390 waiting?” on page 25).

Is a CICS transaction waiting?

Consider the following points:

- CICS could be under stress. This could indicate that the maximum number of tasks allowed (MAXTASK) has been reached, or a short on storage (SOS) condition exists. Check the console log for messages that might explain this (for example, SOS messages), or see the *CICS Problem Determination Guide*.
- The transaction could be waiting for another resource, such as file I/O. You can use CEMT INQ TASK to see what the task is waiting for. If the resource type is MQSERIES your transaction is waiting on MQSeries (either in an **MQGET** WAIT or a task switch). Otherwise see the *CICS Problem Determination Guide* to determine the reason for the wait.
- The transaction could be waiting on MQSeries for OS/390. This could be normal, for example, if your program is a server program that constantly monitors a queue. Otherwise it might be the result of a problem in another program, for example, a transaction abend (see "Messages do not appear on the queue" on page 35). If this is the case, the abend will be reported in the CSMT log.
- If you are using distributed queuing, the program might be waiting for a message that has not yet been delivered from a remote system (for further information, refer to "Problems with missing messages when using distributed queuing" on page 37).

If you suspect that your program has issued an MQI call that did not involve an **MQGET** WAIT (that is, it is in a task switch), and control has not returned from MQSeries, take an SVC dump of both the CICS region, and the MQSeries for OS/390 subsystem before cancelling the CICS transaction. Refer to "Is MQSeries for OS/390 waiting?"

Is MQSeries for OS/390 waiting?

If your enquiries have led you to believe that MQSeries for OS/390 itself is waiting, check the following:

1. Use the DISPLAY THREAD(*) command to check if anything is connected to MQSeries for OS/390 (see the *MQSeries Command Reference* manual for information about the command).
2. Use SDSF DA, or the OS/390 command `DISPLAY A,xxxxMSTR` to determine whether there is any CPU usage (as shown in "Has your application or MQSeries for OS/390 stopped processing work?" on page 17).
   - If MQSeries is using some CPU, you should reconsider other reasons for the perceived wait, or consider whether this is actually a performance problem.
   - If there is no CPU activity, check whether MQSeries for OS/390 will respond to commands. If you can get a response, reconsider other reasons for a perceived wait.
   - If you cannot get a response, check the console log for messages that might explain the wait (for example, MQSeries for OS/390 might have run out of active log data sets, and be waiting for an off-load).

If you are satisfied that MQSeries for OS/390 has actually stalled, cancel it with a dump, and restart. If the problem recurs, refer to Part 5, “Working with IBM to solve your problem” on page 95 for further guidance.

Dealing with loops

The following sections describe the various types of loop that you might encounter, and suggest possible responses.

Is a batch application looping?

If you suspect that a batch or TSO application is looping, use the console to issue the OS/390 command `DISPLAY JOBS,A` (for a batch application) or `DISPLAY TS,A` (for a TSO application). Note the CT values from the data displayed, and repeat the command. If any task shows a significant increase in the CT value, it might be that the task is looping. (You could also use SDSF DA, which shows you the percentage of CPU that each address space is using.)

Is a batch job producing a large amount of output?

An example of this might be an application that browses a queue and prints the messages. If the browse operation has been started with BROWSE FIRST, and subsequent calls have not been reset to BROWSE NEXT, the application will browse and print the first message on the queue repeatedly.

You can use SDSF DA to look at the output of running jobs if you suspect that this is a problem.

Does a CICS region show a lot of CPU activity?

It might be that a CICS application is looping, or that the CICS region itself is in a loop. You might see AICA abends if a transaction goes into a tight (unyielding) loop. If you suspect that CICS, or a CICS application is looping, see the *CICS Problem Determination Guide*.

Does an IMS region show a lot of CPU activity?

It might be that an IMS application is looping. If you suspect this, see the *IMS Diagnosis Guide and Reference* manual.

Is the queue manager showing a lot of CPU activity?

Try to enter an MQSeries DISPLAY command from the console. If you get no response, it is possible that the queue manager is looping. Follow the procedure shown in “Has your application or MQSeries for OS/390 stopped processing work?” on page 17 to display information about the processor time being used by the queue manager. If this indicates that the queue manager is in a loop, cancel it with a dump and restart. If the problem persists, refer to Part 4, “Searching the database for solutions to similar problems” on page 75 and Part 5, “Working with IBM to solve your problem” on page 95 for information on reporting the problem to IBM.

Is a queue or page set filling up unexpectedly?

This could indicate that an application is looping, and putting messages on to a queue. (This could be a batch, CICS, or TSO application.) In a busy system, it might be difficult to identify which application is causing the problem. If you keep a cross-reference of applications to queues, terminate any programs or transactions that could be putting messages on to the queue, and investigate these programs or transactions before using them again. (The most likely culprits will be new, or changed applications; check your change log to identify them.)

It might be that the getting application is at fault, for example, because it has not been triggered.

Using distributed queuing, a symptom of this problem could be a message in the receiving system indicating that **MQPUT** calls to the dead-letter queue are failing. This could be because the dead-letter queue has also filled up. The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code explaining why the message could not be put on to the target queue. See the *MQSeries Application Programming Reference* manual for information about the dead-letter header structure.

If a particular page set frequently fills up, there could be a problem with the allocation of queues to page sets. Refer to “Distribution of queues on page sets” on page 30 for more information.

Are one task, and MQSeries for OS/390, showing a lot of CPU activity?

In this case, a task might be looping on MQI calls (for example, browsing the same message repeatedly).

Chapter 5. Dealing with performance problems

Performance problems are characterized by the following:

- Poor response times in online transactions
- Batch jobs taking a long time to complete

They can be caused by many factors, from a lack of resource in the OS/390 system as a whole, to poor application design. This chapter presents problems and suggested solutions, starting with problems that are relatively simple to diagnose, such as DASD contention, through problems with specific subsystems, such as MQSeries and CICS or IMS, and ending with the more subtle problems of application design, which might take more detecting.

OS/390 system considerations

You might already be aware that your OS/390 system is under stress as these problems will impact many subsystems and applications.

You can use the standard monitoring tools such as Resource Monitoring Facility (RMF) to monitor and diagnose these problems. They might include the following:

- Constraints on storage (paging)
- Constraints on CPU cycles
- Constraints on DASD
- Channel path usage

Normal OS/390 tuning techniques should be used to resolve these problems.

MQSeries for OS/390 considerations

There are a number of decisions to be made when customizing MQSeries for OS/390 that can affect the way your systems perform. These include the following:

- The allocation of data sets
- The allocation of buffers
- The distribution of queues among page sets
- The number of tasks that you allow to access the queue manager at any one time

Log buffer pools

Incorrect allocation of log buffer pools can cause more frequent I/O to logs, which can affect MQSeries for OS/390 performance. RMF reports might show heavy I/O to volumes that hold log data sets.

There are three parameters you can use to tune log buffers. These are specified in the CSQ6LOGP macro (see the *MQSeries for OS/390 System Management Guide* for details), and are:

INBUFF This controls the input buffer size (in the range 28 KB through 60 KB).

OUTBUFF	This controls the size of the output buffer (in the range 40 KB through 4000 KB).
WRTHRSH	This controls the number of buffers to be filled before they are written to the active log data sets (in the range 1 through 256).

You should also be aware of the LOGLOAD parameter of the CSQ6SYSP macro. This specifies the number of log records that are written between checkpoint records. The range is 200 through 16 000 000.

Buffer pool size

There is a buffer pool associated with each page set. The number of buffers in the buffer pool can be specified using the DEFINE BUFFPOOL command. See the *MQSeries Command Reference* manual for more information.

Incorrect specification of buffer pool size can adversely impact MQSeries for OS/390 performance. The smaller the buffer pool, the more frequently physical I/O will be required. RMF might show heavy I/O to volumes that hold page sets.

Distribution of data sets on available DASD

The distribution of page data sets on DASD can have an impact on the performance of MQSeries for OS/390.

Log data sets should be kept on different devices than page data sets (for both performance and system integrity reasons). If you are using dual logging, the logs should also be on different devices, for integrity reasons.

Distribution of queues on page sets

The distribution of queues on page sets can have an impact on performance. This can be indicated by poor response times experienced by transactions using certain queues that reside on heavily used page sets. RMF reports might show heavy I/O to volumes containing the affected page sets.

Queues can be assigned to certain page sets by defining storage class (STGCLASS) objects specifying a particular page set, and then defining the STGCLASS parameter in the queue definition. It is a good idea to define heavily-used queues on different page sets in this way.

Limitation of concurrent threads

The number of tasks accessing the queue manager could also have an impact on performance, particularly if there are other constraints, such as storage, or there are a large number of tasks accessing a few queues. Symptoms of this could be heavy I/O against one or more page sets, or poor response times from tasks known to access the same queues.

There are three parameters specified in the CSQ6SYSP macro that control the number of threads (see the *MQSeries for OS/390 System Management Guide* for details). These are:

IDFORE	Specifies the number of TSO sessions that can connect to the queue manager concurrently.

IDBACK Specifies the number of batch tasks (excluding TSO tasks) that can connect to the queue manager.

CTHREAD Specifies the total number of threads that can connect to the queue manager. This includes batch, TSO, IMS, and CICS.

Notes:

1. Each CICS region takes up nine of the threads specified here, plus one thread for each task initiator (CKTI).
2. The channel initiator makes a number of connections to the queue manager that must be allowed for when setting the CTHREAD system parameter. The number of connections is up to six, plus the number of adapter subtasks (ADAPS), plus the number of dispatchers (DISPS). These values are set in the channel initiator parameter module (CSQ6CHIP); see the *MQSeries for OS/390 System Management Guide* for more information.
3. For IMS connections, the number of threads required is one for the control region, and one for each dependant region connected to MQSeries. For each IMS MPP or IFP region that is defined to permit MQSeries connections through either a specific SSM=EXEC parameter or through the control region default, a thread is created when the first application is scheduled in that region, regardless of whether that application invokes any MQSeries calls. The value you set for CTHREAD should take account of this.

In a CICS environment, you can use CICS MAXTASK to limit concurrent access.

Using the MQSeries trace for administration

Although it will be necessary to use certain traces on occasion, using the trace facility has a negative impact on the performance of your systems.

You should consider what destination you want your trace information sent to. Using the internal trace table will save I/O, but it will not be large enough for traces that produce a lot of data.

The statistics trace gathers information on an interval basis. This interval is controlled by the STATIME parameter of the CSQ6SYSP macro, described in the *MQSeries for OS/390 System Management Guide*.

Traces can be limited by class, resource manager identifier (RMID), and instrumentation facility identifier (IFCID) to reduce the volume of data collected. See the *MQSeries Command Reference* manual for more information.

CICS constraints

Performance of MQSeries tasks can be affected by CICS constraints. For example, your system might have reached MAXTASK, forcing transactions to wait, or the CICS system might be short on storage. If you suspect that CICS is causing your performance problems (for example because batch and TSO jobs run successfully, but your CICS tasks time-out, or have poor response times), see the *CICS Problem Determination Guide* and the *CICS Performance Guide*.

Note: CICS I/O to transient data extrapartition data sets uses OS/390 RESERVE. This could impact I/O to other data sets on the same volume.

When using CICS ISC for distributed queuing, performance problems might be caused by intersystem communications problems, for example:

- You have not defined enough sessions.
- The balance of winners and losers is incorrect (CICS is having to bid to start a conversation).
- You have specified an inappropriate class of service (COS).
- Your system is experiencing networking problems (for example, VTAM® is under stress).

You might suspect that this is a problem if the tasks that are experiencing poor response times are those that are getting messages from a queue that is put to from another system. If you suspect that this is the cause of your problem, see the *CICS Problem Determination Guide* and the *CICS Intercommunication Guide* for further guidance.

Application design considerations

There are a number of ways in which poor program design can affect performance. These can be difficult to detect because the program can appear to perform well, while impacting the performance of other tasks. Several problems specific to programs making MQI calls are discussed in the following sections.

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

Effect of message length

Although MQSeries for OS/390 allows messages to hold up to 4 MB of data, the amount of data in a message affects the performance of the application that processes the message. To achieve the best performance from your application, you should send only the essential data in a message; for example, in a request to debit a bank account, the only information that might need to be passed from the client to the server application is the account number and the amount of the debit.

Effect of message persistence

Persistent messages are logged. Logging messages reduces the performance of your application, so you should use persistent messages for essential data only. If the data in a message can be discarded if the queue manager stops or fails, use a nonpersistent message.

However, the frequency of the logging of persistent messages is determined by the syncpoint and no-syncpoint options when the application puts and gets messages. Messages that are put outside the current unit of work are logged immediately. Messages that are put as part of the current unit of work are logged when the application commits the updates by issuing a syncpoint. Because an application can process many messages between syncpoints, those applications that process messages as part of a unit of work involve fewer logging operations and so have a higher performance.

Searching for a particular message

The **MQGET** call usually retrieves the first message from a queue. If you use the message and correlation identifiers (*MsgId* and *CorrelId*) in the message descriptor to specify a particular message, the queue manager has to search the queue until it finds that message. The use of the **MQGET** call in this way affects the performance of your application because, to find a particular message, MQSeries might have to scan the entire queue.

You can use the *IndexType* queue attribute to specify that you want the queue manager to maintain an index that can be used to increase the speed of **MQGET** operations on the queue. You can choose to build an index of message identifiers or of correlation identifiers, or can choose not to build an index for queues where messages are retrieved sequentially.

You should avoid using this method for queues containing many messages (that is, thousands) as this affects restart time. For a full description of the *IndexType* attribute, see the *MQSeries Application Programming Reference* manual.

Queues that contain messages of different lengths

If the messages on a queue are of different lengths, to determine the size of a message, your application could use the **MQGET** call with the *BufferLength* field set to zero (without MQGMO_ACCEPT_TRUNCATED_MESSAGE) so that, even though the call fails, it returns the size of the message data. The application could then repeat the call, specifying the identifier of the message it measured in its first call and a buffer of the correct size. However, if there are other applications serving the same queue, you might find that the performance of your application is reduced because its second **MQGET** call spends time searching for a message that another application has retrieved in the time between your two calls.

If your application cannot use messages of a fixed length, another solution to this problem is to use the **MQINQ** call to find the maximum size of messages that the queue can accept, then use this value in your **MQGET** call. The maximum size of messages for a queue is stored in the *MaxMsgL* attribute of the queue. This method could use large amounts of storage, however, because the value of this queue attribute could be as high as 4 MB, the maximum allowed by MQSeries for OS/390.

Frequency of syncpoints

Programs that issue a lot of **MQPUT** calls within syncpoint, without committing them, can cause performance problems. Affected queues can fill up with messages that are currently unusable, while other tasks might be waiting to get these messages. This has implications in terms of storage, and in terms of threads tied up with tasks that are attempting to get messages.

You can limit the number of messages that a task can get or put within a single unit of recovery with the DEFINE MAXSMSGS command. See the *MQSeries Command Reference* manual for information about this command.

Advantages of the MQPUT1 call

Use the **MQPUT1** call only if you have a single message to put on a queue. If you want to put more than one message, use the **MQOPEN** call, followed by a series of **MQPUT** calls and a single **MQCLOSE** call.

Chapter 6. Dealing with incorrect output

The term “incorrect output” can be interpreted in many different ways, and its meaning for the purpose of problem determination with this book is explained in Chapter 2, “Identifying the cause of the problem” on page 11.

This chapter discusses the following problems that you could encounter with your system and classify as incorrect output:

- Messages that do not appear when you are expecting them
- Messages that contain the wrong information, or information that has been corrupted

Additional problems that you might encounter if your application includes the use of distributed queues are also discussed.

Messages do not appear on the queue

If messages do not appear when you are expecting them, check for the following:

- Has the message been put onto the queue successfully? Did MQSeries issue a return and reason code for the **MQPUT**, for example:
  - Has the queue been defined correctly, for example is MAXMSGL large enough? (reason code 2030).
  - Are applications able to put messages on to the queue (is the queue enabled for **MQPUT** calls)? (reason code 2051).
  - Is the queue already full? This could mean that an application was unable to put the required message on to the queue (reason code 2053).
- Is the queue a cluster queue? If it is, there might be multiple instances of the queue on different queue managers. Other queue managers can also access a cluster queue.
  - Did you want the message to go to a cluster queue?
  - Is your application designed to work with cluster queues?
  - Did the message get put to a different instance of the queue from that expected?

  Check any cluster-workload exit programs to see that they are processing messages as intended.
- Are you able to get the message from the queue?
  - Do you need to take a syncpoint?

    If messages are being put or got within syncpoint, they are not available to other tasks until the unit of recovery has been committed.
  - Is your timeout interval long enough?
  - Was the message you are expecting defined as persistent?

    If not, and the queue manager has been restarted, the message will have been lost.

  - Are you waiting for a specific message that is identified by a message or correlation identifier (*MsgId* or *CorrelId*)?

    Check that you are waiting for a message with the correct *MsgId* or *CorrelId*. A successful **MQGET** call will set both these values to that of the message got, so you might need to reset these values in order to get another message successfully.

    Also check if you can get other messages from the queue.

  - Can other applications get messages from the queue?

    If so, has another application already retrieved the message?

If you are unable to find anything wrong with the queue, and the queue manager itself is running, make the following checks on the process that you expected to put the message on to the queue:

- Did the application get started?

  If it should have been triggered, check that the correct trigger options were specified.

- Is a trigger monitor running?
- Was the trigger process defined correctly (both to MQSeries for OS/390 and CICS or IMS)?
- Did it complete correctly?

  Look for evidence of an abend, (for example, in the CICS log).

- Did the application commit its changes, or were they backed out?

  Look for messages in the CICS log indicating this.

If multiple transactions are serving the queue, they might occasionally conflict with one another. For example, one transaction might issue an **MQGET** call with a buffer length of zero to find out the length of the message, and then issue a specific **MQGET** call specifying the *MsgId* of that message. However, while this is happening, another transaction might have issued a successful **MQGET** call for that message, so the first application will receive a completion code of MQRC_NO_MSG_AVAILABLE. Applications that are expected to run in a multi-server environment must be designed to cope with this situation.

Have any of your systems suffered an outage? For example, if the message you were expecting should have been put on to the queue by a CICS application, and the CICS system went down, the message might be in doubt. This means that the queue manager does not know whether the message should be committed or backed out, and so has locked it until this is resolved when resynchronization takes place.

Note: The message will be deleted after resynchronization if CICS decides to back it out.

Also consider that the message could have been received, but that your application failed to process it in some way. For example, did an error in the expected format of the message cause your program to reject it? If this is the case, refer to “Messages contain unexpected or corrupted information” on page 40.

Problems with missing messages when using distributed queuing

If your application uses distributed queuing, you should also consider the following points:

- Has distributed queuing been correctly installed on both the sending and receiving systems?

  Ensure that the instructions about installing the distributed queue management facility in the *MQSeries for OS/390 System Management Guide* have been followed correctly.

- Are the links available between the two systems?

  Check that both systems are available, and connected to MQSeries for OS/390. Check that the LU 6.2 or TCP/IP connection between the two systems is active or check the connection definitions on any other systems that you are communicating with.

- Is the channel running?

  – Issue the following command for the transmission queue:

    ```
    DISPLAY QUEUE (qname) IPPROCS
    ```

    If the value for IPPROCS is 0, this means that the channel serving this transmission queue is not running.

  – Issue the following command for the channel:

    ```
    DISPLAY CHSTATUS (channel-name) STATUS MSGS
    ```

    Use the output produced by this command to check that the channel is serving the correct transmission queue and that it is connected to the correct target machine and port. You can determine whether the channel is running from the STATUS field. You can also see if any messages have been sent on the channel by examining the MSGS field.

    If the channel is in RETRYING state, this is probably caused by a problem at the other end. Check that the channel initiator and listener have been started, and that the channel has not been stopped. If somebody has stopped the channel, it will need to be started manually.

  Note: These commands do not apply if you are using CICS ISC for distributed queuing.

- Is triggering set on in the sending system?

  If you are using CICS ISC, check that an instance of CKTI has been started against the initiation queue specified by the transmission queue definition of the sending queue manager. Also check that triggering is set up correctly for this queue.

  If you are not using CICS ISC, check that the channel initiator is running.

- Does the transmission queue have triggering set on?

  If a channel is stopped under certain circumstances, triggering can be set off for the transmission queue.

- Is the message you are waiting for a reply message from a remote system?

  Check the definitions of the remote system, as described above, and check that triggering is activated in the remote system. You should also check that the

LU 6.2 connection between the two systems is not single session (if it is, you will not be able to receive reply messages).

- Is the queue already full?

  This could mean that an application was unable to put the required message on to the queue. If this is so, check if the message has been put on to the dead-letter queue.

  The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code explaining why the message could not be put on to the target queue. See the *MQSeries Application Programming Reference* manual for information about the dead-letter header structure.

- Is there a mismatch between the sending and receiving queue managers?

  For example, the message length could be longer than the receiving queue manager can handle. Check the console log for error messages.

- Are the channel definitions of the sending and receiving channels compatible?

  For example, a mismatch in the sequence number wrap value will stop the channel. See the *MQSeries Intercommunication* manual for more information about distributed queuing.

- Has data conversion been performed correctly?

  If a message has come from a different queue manager, are the CCSIDs and encoding the same, or does data conversion need to be performed.

- Has your channel been defined for fast delivery of nonpersistent messages?

  If your channel has been defined with the NPMSPEED attribute set to FAST (the default), and the channel has stopped for some reason and then been restarted, nonpersistent messages might have been lost. See the *MQSeries Intercommunication* manual for more information about fast messages.

- Is a channel exit causing the messages to be processed in an unexpected way?

  For example, a security exit might prevent a channel from starting, or an *ExitResponse* of MQXCC_CLOSE_CHANNEL might terminate a channel.

Finding messages sent to a cluster queue

Before you can use the techniques described in this chapter to find a message that did not arrive at a cluster queue, you need to determine the queue mangers that host the queue to which the message was sent. You can determine this in the following ways:

- You can use the DISPLAY QUEUE command to request information about cluster queues.

- You can use the name of the queue and queue manager that is returned in the MQPMO structure. If you specified the MQOO_BIND_ON_OPEN option for the message, these fields give the destination of the message. If the message was not bound to a particular queue and queue manager, these fields give the name of the first queue and queue manager to which the message was sent. In this case, this might not be the ultimate destination of the message.

Finding messages sent to the MQSeries-IMS bridge

If you are using the MQSeries-IMS bridge, and your message has not arrived as expected, consider the following:

- Is the MQSeries-IMS bridge running?

  Issue the following command for the bridge queue:

  ```
  DISPLAY QUEUE (qname) IPPROCS CURDEPTH
  ```

  The value of IPPROCS should be 1; if it is 0, check the following:

  – Is the queue a bridge queue?
  – Is IMS running?
  – Has OTMA been started?
  – Is MQSeries connected to OTMA?

  If OTMA is running, the value for CURDEPTH should be 0 because the MQSeries-IMS bridge removes the messages as soon as they arrive on the queue. If the CURDEPTH is greater than 0, check for error messages in the MQSeries job log.

  Use the `/DIS OTMA` command to check that OTMA is active.

- If your messages are flowing to IMS, check the following:

  – Use the `/DIS TMEMBER client TPIPE ALL` command to display information about IMS Tpipes. From this you can determine the number of messages enqueued on, and dequeued from, each Tpipe. (Commit mode 1 messages are not usually queued on a Tpipe.)

  – Use the `/DIS A` command to show whether there is a dependent region available for the IMS transaction to run in.

  – Use the `/DIS TRAN trancode` command to show the number of messages queued for a transaction.

  – Use the `/DIS PROG progname` command to show if a program has been stopped.

- Was the reply message sent to the correct place?

  Issue the following command:

  ```
  DISPLAY QUEUE (*) CURDEPTH
  ```

  Does the CURDEPTH indicate that there is a reply on a queue that you are not expecting?

Messages contain unexpected or corrupted information

If the information contained in the message is not what your application was expecting, or has been corrupted in some way, consider the following points:

- Has your application, or the application that put the message on to the queue changed?

  Ensure that all changes are simultaneously reflected on all systems that need to be aware of the change.

  For example, a copybook formatting the message might have been changed, in which case, both applications will have to be recompiled to pick up the changes. If one application has not been recompiled, the data will appear corrupt to the other.

  You should also check that no external source of data, such as a VSAM data set, has changed. This could also invalidate your data if any necessary recompilations have not been done. Also check that any CICS maps and TSO panels that you are using for input of message data have not changed.

- Is an application sending messages to the wrong queue?

  Check that the messages your application is receiving are not really intended for an application servicing a different queue. If necessary, change your security definitions to prevent unauthorized applications from putting messages on to the wrong queues.

  If your application has used an alias queue, check that the alias points to the correct queue.

  If you altered the queue to make it a cluster queue, it might now contain messages from different application sources.

- Has the trigger information been specified correctly for this queue?

  Check that your application should have been started, or should a different application have been started?

- Has data conversion been performed correctly?

  If a message has come from a different queue manager, are the CCSIDs and encoding the same, or does data conversion need to be performed.

  Check that the *Format* field of the MQMD structure corresponds with the content of the message. If not, the data conversion process might not have been able to deal with the message correctly.

If these checks do not enable you to solve the problem, you should check your application logic, both for the program sending the message, and for the program receiving it.

Part 3. Diagnostic aids and techniques

Chapter 7. Diagnostic aids

This chapter discusses the following subjects:

- The recovery actions attempted by the queue manager when a problem is detected.
- MQSeries for OS/390 abends, and the information produced when an abend occurs.
- The diagnostic information produced by MQSeries for OS/390, and additional sources of useful information.

The type of information provided to help with problem determination and application debugging depends on the type of error encountered, and the way your subsystem is set up.

MQSeries for OS/390 recovery actions

MQSeries for OS/390 can recover from program checks caused by incorrect user data. A completion and reason code is issued to the caller. These codes are documented in the *MQSeries Application Programming Reference* manual.

Program errors

Program errors might be associated with user application program code or MQSeries code, and fall into two categories:

- User-detected errors
- Subsystem-detected errors

User-detected errors

User-detected errors are detected by the user (or a user-written application program) when the results of a service request are not as expected (for example, a nonzero completion code). Because detection occurs after the MQSeries function has completed, the collection of problem determination data cannot be automated. Rerunning the application with the MQSeries for OS/390 user parameter trace facility activated can provide the data needed to analyze the problem. The output from this trace is directed to the *generalized trace facility* (GTF).

The trace can be turned on and off by operator command. Refer to Chapter 9, “Using trace for problem determination” on page 67 for more information.

Queue manager detected errors

The queue manager detects errors such as:

- A program check
- A data set filling up
- An internal consistency error

MQSeries for OS/390 analyzes the error and takes the following actions:

- If the problem was caused by a user or application error (such as an invalid address being used), the error is reflected back to the application by completion and reason codes.

- If the problem was not caused by a user or application error (for example, all available DASD has been used, or the system detected an internal inconsistency), MQSeries for OS/390 recovers if possible (either by sending completion and reason codes to the application, or abending the application if this is not possible).
- If MQSeries for OS/390 is unable to recover, it terminates with a specific reason code. An SVC dump is usually taken recording information in the *system diagnostic work area* (SDWA) and *variable recording area* (VRA) portions of the dump, and an entry is made in SYS1.LOGREC.

MQSeries for OS/390 abends

MQSeries for OS/390 uses two system abend completion codes, `X'5C6'` and `X'6C6'`. These codes identify:

- Internal errors encountered during operation
- Diagnostic information for problem determination
- Actions initiated by the component involved in the error

X'5C6'
A `X'5C6'` abend completion code indicates that MQSeries for OS/390 has detected an internal error and has terminated an internal task (TCB) or a user-connected task abnormally. Errors associated with a `X'5C6'` abend completion code might be preceded by an OS/390 system code, or by internal errors.

The diagnostic material generated by the `X'5C6'` abend should be examined to determine the source of the error that actually resulted in a subsequent task or subsystem termination.

X'6C6'
A `X'6C6'` abend completion code indicates that MQSeries for OS/390 has detected a severe error and has terminated the entire queue manager abnormally. When a `X'6C6'` is issued, MQSeries for OS/390 has determined that continued operation could result in the loss of data integrity. Errors associated with a `X'6C6'` abend completion code might be preceded by an OS/390 system error, one or more `X'5C6'` abend completion codes, or by error message CSQV086E indicating abnormal termination of MQSeries for OS/390.

Table 2 on page 45 summarizes the actions and diagnostic information available to MQSeries for OS/390 when these abend completion codes are issued. Different pieces of this information are relevant in different error situations. The information produced for a given error depends upon the specific problem. The OS/390 services that provide diagnostic information are discussed in “MQSeries for OS/390 diagnostic information” on page 45.

Table 2. Abend completion codes

	`X'5C6'`	`X'6C6'`
Explanation	• Error during MQSeries for OS/390 normal operation	• Severe error; continued operation might jeopardize data integrity
System action	• Internal MQSeries task is abended • Connected user task is abended	• The entire MQSeries subsystem is abended • User task with an active MQSeries connection might be abnormally terminated with a `X'6C6'` code • Possible MEMTERM (memory termination) of connected allied address space
Diagnostic information	• SVC dump • SYS1.LOGREC entry • VRA data entries	• SYS1.LOGREC • VRA data entries
Associated reason codes	• MQSeries abend reason code • Associated OS/390 system codes	• Subsystem termination reason code • OS/390 system completion codes and `X'5C6'` codes that precede the `X'6C6'` abend
Location of accompanying codes	• SVC dump title • Message CSQW050I • Register 15 of SDWA section 'General Purpose Registers at Time of Error' • SYS1.LOGREC entries • VRA data entries	• SYS1.LOGREC • VRA data entries • Message CSQV086E, which is sent to OS/390 system operator

MQSeries for OS/390 diagnostic information

MQSeries for OS/390 functional recovery routines use OS/390 services to provide diagnostic information to help you in problem determination.

The following OS/390 services provide diagnostic information:

- SVC dumps

  The MQSeries for OS/390 abend completion code `X'5C6'` uses the OS/390 SDUMP service to create SVC dumps. The content and storage areas associated with these dumps vary, depending on the specific error and the state of the queue manager at the time the error occurred.

- SYS1.LOGREC

  Entries are requested in the SYS1.LOGREC data set at the time of the error using the OS/390 SETRP service. The following are also recorded in SYS1.LOGREC:

  – Subsystem abnormal terminations

  – Secondary abends occurring in a recovery routine

  – Recording requests from recovery routines percolated to by the recovery termination manager

- Variable recording area (VRA) data

  Data entries are added to the VRA of the SDWA by using an OS/390 VRA defined key. VRA data includes a series of diagnostic data entries common to all MQSeries for OS/390 abend completion codes. Additional information is provided by the invoking component recovery routine during initial error processing or recovery termination manager percolation.

OS/390 abends

During MQSeries operation, an abend might occur with an OS/390 system completion code. If you receive an OS/390 abend, see the appropriate OS/390 publication.

CICS abends

A CICS abend message is sent to the terminal, if the application is attached to one, or to the CSMT log. CICS abend codes are explained in the *CICS Messages and Codes* manual.

The CICS adapter and the distributed queuing (using CICS ISC) component issue abend reason codes beginning with the letter ‘Q’ (for example, QLOP). These codes are documented in the *MQSeries for OS/390 Messages and Codes* manual.

IMS abends

An IMS application might abend in one of the following circumstances:

- A normal abend.
- An IMS pseudo abend, with an abend code such as U3044 resulting from an error in an ESAF exit program.
- Abend 3051 or 3047, when the REO (region error option) has been specified as ‘Q’ or ‘A’, and an IMS application attempts to reference a non-operational external subsystem, or resources are unavailable at create thread time.

An IMS message is sent to the user terminal or job output, and the IMS master terminal. The abend might be accompanied by a region dump.

Diagnostic information produced

MQSeries for OS/390 provides unique messages that, together with the output of dumps, are aimed at providing sufficient data to allow diagnosis of the problem without having to try to reproduce it. This is known as first failure data capture.

Error messages

MQSeries for OS/390 tries to produce an error message when a problem is detected. MQSeries for OS/390 diagnostic messages begin with the prefix CSQ. Each error message generated by MQSeries is unique; that is, it is generated for one and only one error. Information about the error can be found in the *MQSeries for OS/390 Messages and Codes* manual.

The first three characters of the names of MQSeries for OS/390 modules are also usually CSQ. The exceptions to this are modules for the internet gateway (DMQ), C++ (IMQ), and the header files (CMQ). The fourth character uniquely identifies the component. These identifiers are listed in Appendix B, “MQSeries component and resource manager identifiers” on page 109. Characters five through eight are unique within the group identified by the first four characters.

Make sure that you have some documentation on application messages and codes for programs that were written at your installation, as well as a copy of the *MQSeries for OS/390 Messages and Codes* manual.

There might be some instances when no message is produced, or, if one is produced, it cannot be communicated. In these circumstances, you might have to analyze a dump to isolate the error to a particular module. The use of dumps is discussed in Chapter 8, “MQSeries dumps” on page 51.

Dumps

Dumps are an important source of detailed information about problems. Whether they are as the result of an abend or a user request, they allow you to see a “snapshot” of what was happening at the moment the dump was taken. Chapter 8, “MQSeries dumps” on page 51 contains guidance about using dumps to locate problems in your MQSeries system. However, because they do only provide a “snapshot”, you might need to use them in conjunction with other sources of information that cover a longer period of time, such as logs.

Snap dumps are also produced in CSQSNAP for certain types of error in handling MQI calls.

Console logs and job output

Console logs can be copied into a permanent data set, or printed as required. If you are only interested in certain events, you can select which parts of the console log to print.

Job output includes output produced from running the job, as well as that from the console. This can be copied into permanent data sets, or printed as required. You might need to collect output for all associated jobs, for example CICS, IMS, and MQSeries.

Symptom strings

Symptom strings display important diagnostic information in a structured format. When a symptom string is produced, it is available in one or more of the following places:

- On the OS/390 system console
- In SYS1.LOGREC
- In any dump taken

Figure 1 shows an example of a symptom string.

```
PIDS/5655A9500 RIDS/CSQMAIN1 AB/S6C6 PRCS/0E30003
```

Figure 1. Sample symptom string

The symptom string provides a number of keywords that can be directly typed in and used to search the IBM software support database. If you have access to one of the optional search tools, you can search the database yourself. If you report a problem to the IBM support center, you are often asked to quote the symptom string. (For more information about searching the IBM software support database, refer to Part 4, “Searching the database for solutions to similar problems” on page 75.)

Although the symptom string is really designed to provide keywords for searching the database, it can also give you a lot of information about what was happening at the time the error occurred, and it might suggest an obvious cause or a promising area to start your investigation. See Chapter 11, “Building a keyword string” on page 81 for more information about keywords.

Queue information

You can display information about the status of queues by using the operations and control panels, or by entering the DISPLAY QUEUE command from the OS/390 console. See the *MQSeries Command Reference* manual for information about commands and how to issue them.

Note: If the command was issued from the console, the response is copied to the console log, allowing the documentation to be kept together compactly.

Other sources of information

You might find the following items of documentation useful when solving problems with MQSeries for OS/390.

Your own documentation

Your own documentation is the collection of information produced by your organization about what your system and applications should do, and how they are supposed to do it. How much of this kind of information you need depends on how familiar you are with the system or application in question, and could include:

- Program descriptions or functional specifications
- Flowcharts or other descriptions of the flow of activity in a system
- Change history of a program
- Change history of your installation
- Statistical and monitoring profile showing average inputs, outputs, and response times

Manuals for the products you are using

The manuals for the product you are using are the books in the MQSeries library (see “MQSeries publications” on page x) and in the libraries for any other products you use with your application.

Make sure that the level of any book you refer to matches the level of the system you are using. Problems often arise through using either obsolete information, or information about a level of a product that is not yet installed.

Source listings and link-edit maps

Include the source listings of any applications written at your installation with your set of documentation. (They can often be the largest single element of documentation. Large installations with thousands of programs often keep such listings on microfiche.) Make sure you include the relevant linkage editor output with your source listings to avoid wasting time trying to find your way through a load module with an out-of-date link map. Be sure to include the JCL at the beginning of your listings, to show the libraries that were used and the load library the load module was placed in.

Change log

The information in the change log can tell you of changes made in the data processing environment that might have caused problems with your application program. To get the most out of your change log, include the data concerning hardware changes, system software (such as OS/390 and MQSeries) changes, application changes, and any modifications made to operating procedures.

System configuration charts

System configuration charts show what systems are running, where they are running, and how the systems are connected to each other. They also show which MQSeries, CICS, or IMS systems are test systems and which are production systems.

Diagnostic aids for CICS

The CKQC transaction (the CICS adapter control panels) can be used to display information about queue manager tasks, and what state they are in (for example, a GET WAIT). See the *MQSeries for OS/390 System Management Guide* for information about CKQC.

The application development environment is the same as for any other CICS application, and so any tools normally used in that environment can be used to develop MQSeries applications. In particular, the *CICS execution diagnostic facility* (CEDF) traps entry to and exit from the CICS adapter for each MQI call, as well as trapping calls to all CICS API services. Examples of the output produced by this facility are given in Appendix D, “Examples of CEDF output” on page 113.

The CICS adapter also writes trace entries to the CICS trace. These entries are described in Appendix C, “CICS adapter trace entries” on page 111.

Additional trace and dump data is available from the CICS region. These entries are as described in the *CICS Problem Determination Guide*.

Diagnostic aids for IMS

The application development environment is the same as for any other IMS application, and so any tools normally used in that environment can be used to develop MQSeries applications.

Trace and dump data is available from the IMS region. These entries are as described in the *IMS/ESA Diagnosis Guide and Reference* manual.

Chapter 8. MQSeries dumps

This chapter discusses the use of dumps in problem determination. It describes the steps you should take when looking at a dump produced by an MQSeries for OS/390 address space. The following topics are discussed:

How to use dumps for problem determination

When solving problems with your MQSeries for OS/390 system, you can use dumps in two ways:

- To examine the way MQSeries processes a request from an application program.

  To do this, you will usually need to analyze the whole dump, including control blocks and the internal trace.

- To identify problems with MQSeries for OS/390 itself, under the direction of IBM support center personnel.

You will often find that the dump title provides sufficient information in the abend and reason codes to resolve the problem. You can see the dump title in the console log, or by using the OS/390 `DISPLAY DUMP,TITLE` command. The format of the dump title is explained in "Analyzing the dump" on page 62. MQSeries for OS/390 abend codes are discussed in "MQSeries for OS/390 abends" on page 44, and abend reason codes are documented in the *MQSeries for OS/390 Messages and Codes* manual.

If there is not enough information about your problem in the dump title, you will need to format the dump to display the other information contained in it.

Getting a dump

The following table shows information about the types of dump used with MQSeries for OS/390 and how they are initiated. It also shows how the dump is formatted:

Table 3. Types of dump used with MQSeries for OS/390

Dump type	Data set	Output type	Formatted by	Caused by
SVC	Defined by system	Machine readable	IPCS in conjunction with an MQSeries for OS/390 verb exit	OS/390 or MQSeries for OS/390 functional recovery routine detecting error, or the operator entering the OS/390 DUMP command
SYSUDUMP	Defined by JCL (SYSOUT=A)	Formatted	Normally SYSOUT=A	An abend condition (only taken if there is a SYSUDUMP DD statement for the step)
Snap	Defined by JCL CSQSNAP (SYSOUT=A)	Formatted	Normally SYSOUT=A	Unexpected MQI call errors reported to adapters, or FFST™ information from the channel initiator
Stand-alone	Defined by installation (tape or disk)	Machine readable	IPCS in conjunction with an MQSeries for OS/390 verb exit	Operator IPL of the stand-alone dump program

MQSeries for OS/390 recovery routines request SVC dumps for most `X'5C6'` abends. The exceptions are listed in “When SVC dumps are not produced” on page 66. SVC dumps issued by MQSeries for OS/390 are the primary source of diagnostic information for problems.

If the dump is initiated by the MQSeries for OS/390 subsystem, information about the dump is put into area called the *summary portion*. This contains information that the dump formatting program can use to identify the key components.

For more information about SVC dumps, see the *MVS Diagnosis: Tools and Service Aids* manual.

Using the OS/390 DUMP command

You might be asked to take a dump of the main MQSeries address space or the channel initiator address space for IBM to resolve a problem. Figure 2 on page 53 and Figure 3 on page 53 show examples of the OS/390 commands to do this, assuming a subsystem name of CSQ1.

```
DUMP COMM=(MQSERIES MAIN DUMP)
*01 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R 01,JOBNAME=CSQ1MSTR,CONT
*02 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 01 IS;JOBNAME=CSQ1MSTR,CONT
R 02,SDATA=(CSA,RGN,PSA,SQA,TRT),END
 IEE600I REPLY TO 02 IS;SDATA=(CSA,RGN,PSA,SQA,TRT),END
IEA794I SVC DUMP HAS CAPTURED: 869
DUMPID=001 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=MQSERIES MAIN DUMP
```

Figure 2. Example of taking a dump of the MQSeries address space

```
DUMP COMM=(MQSERIES CHIN DUMP)
*01 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R 01,JOBNAME=CSQ1CHIN,CONT
*02 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 01 IS;JOBNAME=CSQ1CHIN,CONT
R 02,SDATA=(CSA,RGN,PSA,SQA,TRT),CONT
*03 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 02 IS;SDATA=(CSA,RGN,PSA,SQA,TRT),CONT
R 03,DSPNAME='CSQ1CHIN'.CSQXTRDS,END
IEE600I REPLY TO 03 IS;DSPNAME='CSQ1CHIN'.CSQXTRDS,END
IEA794I SVC DUMP HAS CAPTURED: 869
DUMPID=001 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=MQSERIES CHIN DUMP
```

Figure 3. Example of taking a dump of the channel initiator address space

Processing a dump

You can process a dump in several ways:

- Use the IPCS panels provided by MQSeries for OS/390, as described in “Using the MQSeries for OS/390 dump display panels” on page 54.
- Use the IPCS dialog under TSO using line mode IPCS commands, as described in “Using line mode IPCS” on page 59
- Use IPCS as a batch job to produce the output as a print file, as described in “Using IPCS in batch” on page 61

Using the MQSeries for OS/390 dump display panels

MQSeries for OS/390 provides a set of panels to help you process dumps. The following section describes how to use these panels:

1. From the IPCS PRIMARY OPTION MENU, select **ANALYSIS - Analyze dump contents** (option 2).

   The IPCS MVS ANALYSIS OF DUMP CONTENTS panel appears.

2. Select **COMPONENT - MVS component data** (option 6).

   The IPCS MVS DUMP COMPONENT DATA ANALYSIS panel appears. The appearance of the panel depends on the products installed at your installation, but will be similar to the panel shown below:

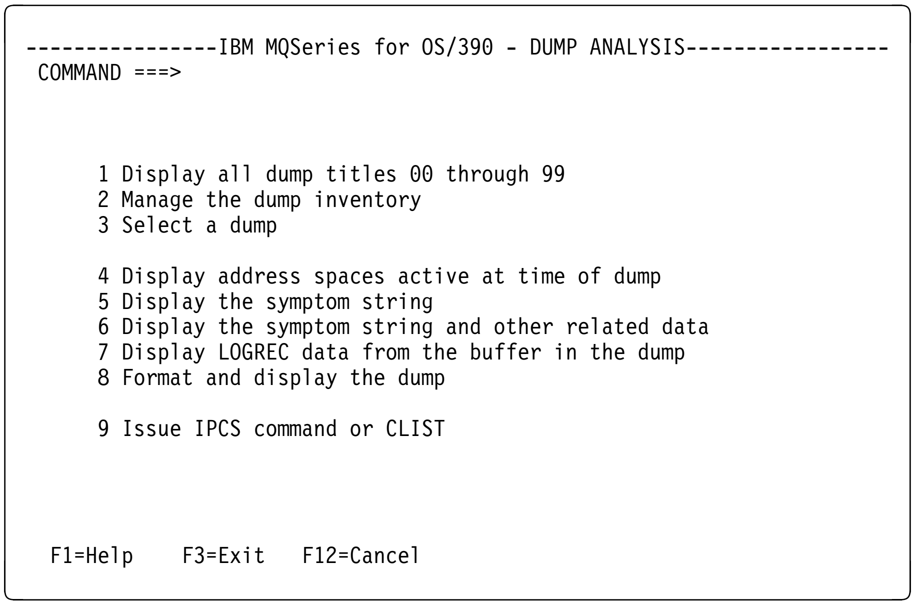
```
---------------- IPCS MVS DUMP COMPONENT DATA ANALYSIS -------------
OPTION ===>                                              SCROLL ===

To display information, specify "S option name" or enter S to the
left of the option desired.  Enter ? to the left of an option to
display help regarding the component support.

  Name     Abstract
  ALCWAIT  Allocation wait summary
  AOMDATA  AOM analysis
  ASMCHECK Auxiliary storage paging activity
  ASMDATA  ASM control block analysis
  AVMDATA  AVM control block analysis
  COMCHECK Operator communications data
  CSQMAIN  MQSeries dump formatter panel interface
  CSQWDMP  MQSeries dump formatter
  CTRACE   Component trace summary
  DAEDATA  DAE header data
  DIVDATA  Data-in-virtual storage
```

3. Select **CSQMAIN MQSeries dump formatter panel interface** by typing **s** beside the line and pressing Enter.

   If this option is not available, it is because the member CSQ7IPCS is not present; you should see the *MQSeries for OS/390 System Management Guide* for information about installing the MQSeries for OS/390 dump formatting member.

   Note: If you have already used the dump to do some preliminary analysis, and you wish to re-examine it, select **CSQWDMP MQSeries dump formatter** to re-display the formatted contents, using the default options.

4. The **IBM MQSeries for OS/390 - DUMP ANALYSIS** menu appears. Use this menu to specify the action that you want to perform on a system dump.



```
----------------IBM MQSeries for OS/390 - DUMP ANALYSIS-----------------
 COMMAND ===>



      1 Display all dump titles 00 through 99
      2 Manage the dump inventory
      3 Select a dump

      4 Display address spaces active at time of dump
      5 Display the symptom string
      6 Display the symptom string and other related data
      7 Display LOGREC data from the buffer in the dump
      8 Format and display the dump

      9 Issue IPCS command or CLIST




  F1=Help    F3=Exit   F12=Cancel
```

5. Before you can select a particular dump for analysis, the dump you require must be present in the dump inventory. To ensure that this is so, perform the following steps:

   a. If you do not know the name of the data set containing the dump, specify option 1 - **Display all dump titles xx through xx**.

      This displays the dump titles of all the dumps contained in the SYS1.DUMP data sets (where xx is a number in the range 00 through 99). You can limit the selection of data sets for display by using the xx fields to specify a range of data set numbers.

      If you want to see details of all available dump data sets, set these values to 00 and 99.

      Use the information displayed to identify the dump you want to analyze.

   b. If the dump has not been copied into another data set (that is, it is in one of the SYS1.DUMP data sets), specify option 2 - **Manage the dump inventory**

      The dump inventory contains the dump data sets that you have used. Because the SYS1.DUMP data sets are reused, the name of the dump that you identified in step 5a might appear in the list displayed. However, this entry refers to the previous dump that was stored in this data set, so delete it by typing DD next to it and pressing Enter. Then press F3 to return to the DUMP ANALYSIS MENU.

6. Specify option 3 - **Select a dump**, to select the dump that you want to work with. Type the name of the data set containing the dump in the Source field, check that NOPRINT and TERMINAL are specified in the Message Routing field (this is to ensure that the output is directed to the terminal), and press Enter. Press F3 to return to the DUMP ANALYSIS MENU.

7. Having selected a dump to work with, you can now use the other options on the menu to analyze the data in different parts of the dump:

   - To display a list of all address spaces active at the time the dump was taken, select option 4.
   - To display the symptom string, select option 5. If you want to use the symptom string to search the RETAIN® database for solutions to similar problems, refer to Chapter 10, “Searching the database” on page 77.
   - To display the symptom string and other serviceability information, including the variable recording area of the system diagnostic work area (SDWA), select option 6.
   - To format and display the data contained in the in-storage LOGREC buffer, select option 7.

     It could be that the abend that caused the dump was not the original cause of the error, but was caused by an earlier problem. To determine which LOGREC record relates to the cause of the problem, go to the bottom of the data set, type `FIND ERRORID: PREV`, and press Enter. The header of the latest LOGREC record is displayed, for example:

     ```
     JOBNAME: NONE-FRR
     ERRORID: SEQ=00081  CPU=0040  ASID=0033  TIME=14:42:47.1

     SEARCH ARGUMENT ABSTRACT

       PIDS/5655A9500 RIDS/CSQRLLM1#L RIDS/CSQRRHSL AB/S05C6
       PRCS/00D10231 REGS/0C1F0 RIDS/CSQVEUS2#R

       SYMPTOM            DESCRIPTION
       -------            -----------
       PIDS/5655A9500     PROGRAM ID: 5655A9500
       .
       .
       .
     ```

     Note the program identifier (if it is not 5655A9500, the problem was not caused by MQSeries for OS/390 and you could be looking at the wrong dump). Also note the value of the TIME field. Repeat the command to find the previous LOGREC record, and note the value of the TIME field again. If the two values are close to each other (say, within about one or two tenths of a second), they could both relate to the same problem.

     You can use the symptom string from the LOGREC record related to the error to search the RETAIN database for solutions to similar problems (refer to Chapter 10, “Searching the database” on page 77).

- To format and display the dump, select option 8. The **FORMAT AND DISPLAY THE DUMP** panel appears:

```
----------IBM MQSeries for OS/390 - FORMAT AND DISPLAY DUMP---------
 COMMAND ===>

    1  Display the control blocks and trace
    2  Display just the control blocks
    3  Display just the trace


 Options:

 Use the summary dump? . . . . . . . . . . . . . .  __ 1 Yes
                                                       2 No


 Subsystem name (required if summary dump not used) ____


 Address space identifier or ALL. . . . . . . . . . ALL_



  F1=Help    F3=Exit   F12=Cancel
```

- Use this panel to format your selected system dump. You can choose to display control blocks, data produced by the internal trace, or both (the default).

  Note: You can't do this for dumps from the channel initiator.

  – To display the whole of the dump, that is:

    - The dump title
    - The variable recording area (VRA) diagnostic information report
    - The save area trace report
    - The control block summary
    - The trace table

    select option 1.

  – To display the information listed for option 1, without the trace table, select option 2.

  – To display the information listed for option 1, without the control blocks, select option 3.

  You can also use the following options:

  – **Use the Summary Dump?**

    Use this field to specify whether you want MQSeries for OS/390 to use the information contained in the summary portion when formatting the selected dump. The default setting is YES.

    Note: If a summary dump has been taken, it might include data from more than one address space.

  – **Subsystem name**

    Use this field to identify the subsystem whose dump data you want to display. This is only required if there is no summary data (for example, if the operator requested the dump), or if you have specified NO in the **Use the summary dump?** field.

If you do not know the subsystem name, type `IPCS SELECT ALL ERROR` at the command prompt, and press Enter to display a list of all the jobs running at the time of the error. Note the name of the job that has the word ERROR against it in the SELECTION CRITERIA column. It is of the form *xxxx*MSTR, where *xxxx* is the subsystem name.

```
IPCS OUTPUT STREAM -------------------------
COMMAND ===>
  ASID JOBNAME  ASCBADDR  SELECTION CRITERIA
  ---- -------- --------  ------------------
  0001 *MASTER* 00FD4D80  ALL
  0002 PCAUTH   00F8AB80  ALL
  0003 RASP     00F8C100  ALL
  0004 TRACE    00F8BE00  ALL
  0005 GRS      00F8BC00  ALL
  0006 DUMPSRV  00F8DE00  ALL
  0008 CONSOLE  00FA7E00  ALL
  0009 ALLOCAS  00F8D780  ALL
  000A SMF      00FA4A00  ALL
  000B VLF      00FA4800  ALL
  000C LLA      00FA4600  ALL
  000D JESM     00F71E00  ALL
  001F MQM1MSTR 00FA0680  ERROR ALL
```

If no job has the word ERROR against it in the SELECTION CRITERIA column, type IPCS BLSCSETD at the command prompt, and press Enter to display the IPCS Default Values panel. Note the address space identifier (ASID) and press F3 to return to the previous panel. Use the ASID to determine the job name; it is of the form *xxxx*MSTR, where *xxxx* is the subsystem name.

Press F3 to return to the FORMAT AND DISPLAY THE DUMP panel, and type this name in the **Subsystem name** field.

– **Address space identifier**

Use this field if the data in a dump comes from more than one address space. If you only want to look at data from a particular address space, specify the identifier (ASID) for that address space.

The default value for this field is ALL, which causes information about all the address spaces relevant to the subsystem in the dump to be displayed. Change this field by typing the 4-character ASID over the value displayed.

Note: Because the dump contains storage areas common to all address spaces, the information displayed might not be relevant to your problem if you specify the address space identifier incorrectly. In this case, return to this panel, and enter the correct address space identifier.

Using line mode IPCS

To format the dump using line mode IPCS commands, select the dump required by issuing the

```
SETDEF DSN('SYS1.DUMPxx')
```

command (where `SYS1.DUMPxx` is the name of the data set containing the dump). You can then use IPCS subcommands to display data from the dump.

The IPCS VERBEXIT CSQWDMP invokes the MQSeries for OS/390 dump formatting program (CSQWDPRD), and enables you to format an SVC dump to display MQSeries for OS/390 data. You can restrict the amount of data that is displayed by specifying parameters.

Note: You can't do this for dumps from the channel initiator.

This section describes the parameters required to extract the necessary data. You should separate operands by commas, not blanks. A blank that follows any operand in the control statement terminates the operand list, and any subsequent operands are ignored. Table 4 lists and explains each of the various keywords you can specify in the control statement for formatting dumps.

Table 4. Keywords for the MQSeries for OS/390 dump formatting control statement

Keyword	Effect
ALL	All of the control blocks and the trace table are formatted (this is the default).
TT	The trace table only is formatted.
SG	A few key system-wide control blocks are formatted.
LG	All of the control blocks are formatted.
AA	Data is displayed for all MQSeries for OS/390 control blocks in all address spaces.
SA=*hhhh*	Only the control blocks for a specified address space are displayed (the format of this keyword can be SA=*hh* or SA=*hhhh*, where *h* represents a hexadecimal digit).
EB=*nnnnnnnn*	Only the trace points associated with this EB thread are displayed (the format of this keyword is EB=*nnnnnnnn* where *nnnnnnnn* is the 8-digit address of an EB thread that is contained in the trace). This must be used in conjunction with the TT keyword.

If the dump is initiated by the operator, then there is no information in the summary portion of the dump. Table 5 shows additional keywords that can be used in the CSQWDMP control statement.

Table 5. Summary dump keywords for the MQSeries for OS/390 dump formatting control statement

Keyword	Description
SUBSYS=*aaaa*	Use this keyword if the summary dump portion is not available, or not to be used, to give the name of the subsystem to format information for. *aaaa* is a 1 through 4-character subsystem name.
SUMDUMP=NO	Use this keyword if the dump has a summary portion, but you do not want to use it. (You would usually only do this if so directed by your IBM support center.)

The following list shows some examples of how to use of these keywords:

- For default formatting of all address spaces, using information from the summary portion of the dump use:

  ```
  VERBX CSQWDMP
  ```

- To display the trace table from a dump of subsystem named MQMT, which was initiated by an operator (and so does not have a summary portion) use:

  ```
  VERBX CSQWDMP 'TT,SUBSYS=MQMT'
  ```

- To display all the control blocks and the trace table from a dump produced by a subsystem abend, for an address space with ASID (address space identifier) 1F, use:

  ```
  VERBX CSQWDMP 'TT,LG,SA=1F'
  ```

- To display the portion of the trace table from a dump associated with a particular EB thread, use:

  ```
  VERBX CSQWDMP 'TT,EB=nnnnnnnn'
  ```

Table 6 shows some other commands that are used frequently for analyzing dumps. For more information about these subcommands, see the *MVS IPCS Commands* manual.

Table 6. IPCS subcommands used for dump analysis

Subcommand	Description
STATUS	To display data usually examined during the initial part of the problem determination process.
VERBEXIT LOGDATA	To format the in-storage LOGREC buffer records present before the dump was taken. LOGDATA locates the LOGREC entries that are contained in the LOGREC recording buffer and invokes the EREP program to format and print the LOGREC entries. These entries are formatted in the style of the normal detail edit report.
VERBEXIT TRACE	To format the system trace entries for all address spaces.
VERBEXIT SYMPTOM	To format the symptom strings contained in the header record of a system dump such as stand-alone dump, SVC dump, or an abend dump requested with a SYSUDUMP DD statement.
VERBEXIT GRSTRACE	To format diagnostic data from the major control blocks for global resource serialization.
VERBEXIT SUMDUMP	To locate and display the summary dump data that an SVC dump provides.
VERBEXIT DAEDATA	To format the dump analysis and elimination (DAE) data for the dumped system.

Using IPCS in batch

To use IPCS in batch, insert the required IPCS statements into your batch job stream (see Figure 4).

Change the data set name (DSN=) on the DUMP00 statement to reflect the dump you want to process, and insert the IPCS subcommands that you want to use.

```
//*************************************************
//*   RUNNING IPCS IN A BATCH JOB                 *
//*************************************************
//MQMDMP   EXEC  PGM=IKJEFT01,REGION=5120K
//STEPLIB  DD    DSN=mqm.library-name,DISP=SHR
//SYSTSPRT DD    SYSOUT=*
//IPCSPRNT DD    SYSOUT=*
//IPCSDDIR DD    DSN=dump.directory-name,DISP=OLD
//DUMP00   DD    DSN=dump.name,DISP=SHR
//SYSTSIN  DD    *
  IPCS NOPARM TASKLIB(SCSQLOAD)
  SETDEF PRINT TERMINAL DDNAME(DUMP00) NOCONFIRM
  **************************************************
  * INSERT YOUR IPCS COMMANDS HERE, FOR EXAMPLE:   *
  VERBEXIT LOGDATA
  VERBEXIT SYMPTOM
  VERBEXIT CSQWDMP 'TT,SUBSYS=QMGR'
  **************************************************

  CLOSE   ALL
END
/*
```

Figure 4. Sample JCL for printing dumps through IPCS in the OS/390 environment

Analyzing the dump

The dump title includes the abend completion and reason codes, the failing load module and CSECT names, and the release identifier.

The formats of SVC dump titles vary slightly, depending on the type of error.

Figure 5 shows an example of an SVC dump title. Each field in the title is described after the figure.

```
ssnm,ABN=5C6-00D303F2,U=AUSER,C=A9500.210.LOCK-CSQL1GET,
 M=CSQGFRCV,LOC=CSQLLPLM.CSQL1GET+0246
```

Figure 5. Sample SVC dump title

`ssnm,ABN=compltn-reason`

: `ssnm` is the name of the subsystem that issued the dump.

  `compltn` is the 3-character hexadecimal abend completion code (in this example, X'5C6'), prefixed by U for user abend codes.

  `reason` is the 4-byte hexadecimal reason code (in this example, X'00D303F2').

Note: The abend and reason codes might provide sufficient information to resolve the problem. See the *MQSeries for OS/390 Messages and Codes* manual for an explanation of the reason code.

`U=userid`

: `userid` is the user identifier of the user (in this example, AUSER). This field is not present for channel initiators.

`C=compid.release.comp-function`

: `compid` is the last 5 characters of the component identifier (explained in “Component-identifier keyword” on page 81). The value A9500 uniquely identifies MQSeries for OS/390.

  `release` is a 3-digit code indicating the version, release, and modification level of MQSeries for OS/390 (in this example, 210).

  `comp` is an acronym for the component in control at the time of the abend (in this example, LOCK).

  `function` is the name of a function, macro, or routine in control at the time of abend (in this example, CSQL1GET). This field is not always present.

`M=module`

: `module` is the name of the FRR or ESTAE recovery routine (in this example, CSQGFRCV). This field is not always present.

  Note: This is not the name of the module where the abend occurred; that is given by `LOC`.

`LOC=loadmod.csect+csect_offset`

`loadmod` is the name of the load module in control at the time of the abend (in this example, CSQLLPLM). This might be represented by an asterisk if it is unknown.

`csect` is the name of the CSECT in control at the time of abend (in this example, CSQL1GET).

`csect_offset` is the offset within the failing CSECT at the time of abend (in this example, 0246).

Note: The value of `csect_offset` might vary if service has been applied to this CSECT, so this value should not be used when building a keyword string to search the IBM software support database.

Dump title variation with PSW and ASID

Some dump titles replace the load module name, CSECT name, and CSECT offset with the PSW (program status word) and ASID (address space identifier). Figure 6 illustrates this format.

```
ssnm,ABN=compltn-reason,U=userid,C=compid.release.comp-function,
 M=module,PSW=psw_contents,ASID=address_space_id
```

Figure 6. Dump title with PSW and ASID

`psw_contents`

This contains the PSW at time of error (in this example, X'077C100000729F9C').

`address_space_id`

This identifies the address space in control at time of abend (in this example, X'0011'). This field is not present for a channel initiator.

SYSUDUMP information

SYSUDUMP dumps provide information useful for debugging batch and TSO application programs. For more information about SYSUDUMP dumps, see the *MVS Diagnosis: Tools and Service Aids* manual.

Figure 7 shows a sample of the beginning of a SYSUDUMP dump.

```
JOB MQMBXBA1   STEP TSOUSER   TIME 102912   DATE 97106   ID = 000  CPUID = 632202333081   PAGE 00000001

COMPLETION CODE      SYSTEM = 0C1      REASON CODE = 00000001

PSW AT ENTRY TO ABEND  078D1000 000433FC        ILC 2  INTC 000D

PSW LOAD MODULE = BXBAAB01  ADDRESS = 000433FC  OFFSET = 0000A7F4

ASCB: 00F56400
+0000  ASCB..... ASCB      FWDP..... 00F60180  BWDP..... 0047800  CMSF..... 019D5A30  SVRB..... 008FE9E0
+0014  SYNC..... 00000D6F  IOSP..... 00000000  TNEW..... 00D18F0  CPUS..... 00000001  ASID..... 0066
+0026  R026..... 0000      LL5...... 00        HLHI..... 01       DPHI..... 00        DP....... 9D
+002C  TRQP..... 80F5D381  LDA...... 7FF154E8  RSMF..... 00       R035..... 0000      TRQI..... 42
+0038  CSCB..... 00F4D048  TSB...... 00B61938  EJST..... 0000001  8C257E00

+0048  EWST..... 9CCDE747  76A09480            JSTL..... 00141A4  ECB...... 808FEF78  UBET..... 9CCDE740
  .
  .
  .
ASSB: 01946600
+0000  ASSB..... ASSB      VAFN..... 00000000  EVST..... 0000000  00000000

+0010  VFAT..... 00000000  00000000            RSV...... 000      XMCC..... 0000      XMCT.....00000000
+0020  VSC...... 00000000  NVSC..... 0000004C  ASRR..... 0000000  R02C..... 00000000  00000000 00000000
+0038            00000000  00000000

*** ADDRESS SPACE SWITCH EVENT MASK OFF (ASTESSEM = 0) ***

TCB: 008D18F0
+0000  RBP...... 008FE7D8  PIE...... 00000000  DEB...... 00B1530  TIO...... 008D4000  CMP......805C6000
+0014  TRN...... 40000000  MSS...... 7FFF7418  PKF...... 80       FLGS..... 01000000  00
+0022  LMP...... FF        DSP...... FE        LLS...... 00D1A88  JLB...... 00011F18  JPQ......00000000
+0030  GPR0-3... 00001000  008A4000  00000000  00000000
+0040  GPR4-7... 00FDC730  008A50C8  00000002  80E73F04
+0050  GPR8-11.. 81CC4360  008A6754  008A67B4  00000008
```

Figure 7. Sample beginning of a SYSUDUMP

Snap dumps

Snap dumps are always sent to the data set defined by the CSQSNAP DD statement. They can be issued by the adapters or the channel initiator.

- Snap dumps are issued by the batch, CICS, IMS, or RRS adapter when an unexpected error is returned by the queue manager for an MQI call. A full dump is produced containing information about the program that caused the problem.

  For a snap dump to be produced, the CSQSNAP DD statement must be in the batch application JCL, CICS JCL, or IMS dependent region JCL.

- Snap dumps are issued by the channel initiator in certain error conditions instead of a system dump. The dump contains information relating to the error. Message CSQX053E is also issued at the same time.

  For a snap dump to be produced, the CSQSNAP DD statement must be in the channel initiator started-task procedure.

SYS1.LOGREC information

The SYS1.LOGREC data set records various errors that different components of the operating system encounter. For more information about using SYS1.LOGREC records, see the *MVS Diagnosis: Tools and Service Aids* manual.

MQSeries for OS/390 recovery routines write information in the *system diagnostic work area* (SDWA) to the SYS1.LOGREC data set when retry is attempted, or when percolation to the next recovery routine occurs. Because two or more retries or percolations can occur for a single error, multiple SYS1.LOGREC entries might be recorded.

The SYS1.LOGREC entries recorded near the time of abend might provide valuable historical information about the events leading up to the abend.

Finding the applicable SYS1.LOGREC information

To obtain a SYS1.LOGREC listing, either:

- Use the EREP service aid, described in the *MVS Diagnosis: Tools and Service Aids* manual to format records in the SYS1.LOGREC data set.
- Specify the VERBEXIT LOGDATA keyword in IPCS.
- Use option 7 on the DUMP ANALYSIS MENU (refer to “Using the MQSeries for OS/390 dump display panels” on page 54).

Only records available in storage when the dump was requested are included. Each formatted record follows the heading `*****LOGDATA*****`.

When SVC dumps are not produced

Under some circumstances, SVC dumps are not produced. Generally, dumps are suppressed because of time or space problems, or security violations. The list below summarizes other reasons why SVC dumps might not be produced:

- The OS/390 *serviceability level indication processing* (SLIP) commands suppressed the abend.

  The description of IEACMD00 in the *MVS Initialization and Tuning Reference* manual lists the defaults for SLIP commands executed at IPL time.
- The abend reason code was one which does not require a dump to determine the cause of abend.
- SDWACOMU or SDWAEAS (part of the system diagnostic work area, SDWA) was used to suppress the dump.

Suppressing MQSeries for OS/390 dumps using OS/390 DAE

You can suppress SVC dumps that duplicate previous dumps. The *MVS Diagnosis: Tools and Service Aids* manual gives details about using OS/390 *dump analysis and elimination* (DAE).

To support DAE, MQSeries for OS/390 defines two *variable recording area* (VRA) keys and a minimum symptom string. The two VRA keys are:

- KEY VRADAE (X'53'). No data is associated with this key.
- KEY VRAMINSC (X'52') DATA (X'08')

MQSeries for OS/390 provides the following data for the minimum symptom string in the *system diagnostic work area* (SDWA):

- Load module name
- CSECT name
- Abend code
- Recovery routine name
- Failing instruction area
- REG/PSW difference
- Reason code
- Component identifier
- Component subfunction

Dumps are considered duplicates for purposes of duplicate dump suppression if eight (the X'08' from the VRAMINSC key) of the nine symptoms are the same.

Chapter 9. Using trace for problem determination

The trace facilities available with MQSeries for OS/390 are:

- The user parameter (or API) trace
- The channel initiator trace
- The IBM internal trace used by the support center

This chapter describes how to collect and interpret the data produced by the user parameter trace. The other trace facilities that you can use with MQSeries are also discussed.

The user parameter trace

You can obtain information about MQI calls and user parameters passed by some MQSeries calls on entry to, and exit from, MQSeries. To do this, you should use the global trace in conjunction with the OS/390 generalized trace facility (GTF).

Starting the trace

To use the trace for problem determination, you must start the following:

- The GTF for your OS/390 system
- The MQSeries trace for each queue manager subsystem for which you want to collect data

Starting the GTF

When you start the GTF, you should specify the USRP option. You will be prompted to enter a list of event identifiers (EIDs). The EIDs used by MQSeries are:

5E9	To collect information about control blocks on entry to MQSeries
5EA	To collect information about control blocks on exit from MQSeries

You can also use the JOBNAMEP option, specifying the batch, CICS, IMS, or TSO job name, to limit the trace output to certain jobs. Figure 8 on page 68 illustrates sample startup for the GTF, specifying the two EIDs, and a jobname. The lines shown in bold type **like this** are the commands that you should enter at the console; the other lines are prompts and responses.

For more information about starting the GTF trace, see the *MVS Diagnosis: Tools and Service Aids* manual.

```
 START GTFxx.xx
 £HASP100 GTFxx.xx  ON STCINRDR
 £HASP373 GTFxx.xx  STARTED
*01 AHL100A SPECIFY TRACE OPTIONS
 R 01,TRACE=JOBNAMEP,USRP
 TRACE=JOBNAMEP,USRP
 IEE600I REPLY TO 12 IS;TRACE=JOBNAMEP,USRP
*02 ALH101A SPECIFY TRACE EVENT KEYWORDS - JOBNAME=,USR=
 R 02,JOBNAME=jobname,USR=(5E9,5EA)
 JOBNAME=jobname,USR=(5E9,5EA)
 IEE600I REPLY TO 13 IS;JOBNAME=jobname,USR=(5E9,5EA)
*03 ALH102A CONTINUE TRACE DEFINITION OR REPLY END
 R 03,END
 END
 IEE600I REPLY TO 14 IS;END
 AHL103I TRACE OPTIONS SELECTED-USR=(5EA,5E9)
 AHL103I JOBNAME=(jobname)
*04 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
 R 04,U
 U
 IEE600I REPLY TO 15 IS;U
 AHL031I GTF INITIALIZATION COMPLETE
```

Figure 8. Example startup of GTF to use with the MQSeries trace

Enabling the trace within MQSeries

Use the START TRACE command, specifying type GLOBAL to start writing MQSeries records to the GTF. To define the events that you want to produce trace data for, use one or more of the following classes:

CLASS	Event traced
2	Record the MQI call and MQI parameters when a completion code other than MQRC_NONE is detected.
3	Record the MQI call and MQI parameters on entry to and exit from the queue manager.

Once started, you can display information about, alter the properties of, and stop, the trace with the following commands:

DISPLAY TRACE
ALTER TRACE
STOP TRACE

To use any of the trace commands, you must have one of the following:

- Authority to issue start and stop trace commands (trace authority)
- Authority to issue the display trace command (display authority)

Notes:

1. The trace commands can also be entered through the CSQINP2 initialization input data set.
2. The trace information produced will also include details of syncpoint flows - for example PREPARE and COMMIT.

For information about these commands, see the *MQSeries Command Reference* manual.

Formatting the information

To format the user parameter data collected by the global trace, use either the batch job shown in Figure 9 or the IPCS `GTFTRACE USR(`*xxx*`)` command, where *xxx* is:

5E9	To format information about control blocks on entry to MQSeries MQI calls
5EA	To format information about control blocks on exit from MQSeries MQI calls
5E9,5EA	To format information about control blocks on entry to and exit from MQSeries MQI calls

You can also specify the `JOBNAME(`*jobname*`)` parameter to limit the formatted output to certain jobs.

```
//S1 EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=4096K
//IPCSPARM DD DSN=SYS1.PARMLIB,DISP=SHR
//IPCSDDIR DD DSN=thlqual.ipcs.dataset.directory,DISP=SHR
//SYSTSPRT DD SYSOUT=*,DCB=(LRECL=137)
//IPCSTOC  DD SYSOUT=*
//GTFIN  DD  DSN=gtf.trace,DISP=SHR
//SYSTSIN DD *
IPCS
SETDEF FILE(GTFIN) NOCONFIRM
GTFTRACE USR(5E9,5EA)
/*
//STEPLIB  DD  DSN=thlqual.SCSQAUTH,DISP=SHR
```

Figure 9. Formatting the GTF output in batch. thlqual is your high level qualifier for MQSeries data sets, and gtf.trace is the name of the data set containing your trace information. You must also specify your IPCS data set directory.

Identifying the control blocks associated with MQSeries

The format identifier for the MQSeries trace is D9. This value appears at the beginning of each formatted control block in the formatted GTF output, in the form:

```
USRD9
```

Identifying the event identifier associated with the control block

The trace formatter inserts one of the following messages at the top of each control block. These indicate whether the data was captured on entry to or exit from MQSeries

- CSQW072I ENTRY: MQSeries user parameter trace
- CSQW073I EXIT: MQSeries user parameter trace

If trace data is not produced

If trace data is not produced, check the following:

- Was the GTF started correctly, specifying EIDs 5E9 and 5EA on the USRP option?
- Was the START TRACE(GLOBAL) command entered correctly, and were the relevant classes specified?

Interpreting the information

When you look at the data produced by the GTFTRACE command, you should consider the following points:

- If the control block consists completely of zeros, it is possible that an error occurred while copying data from the users address space. This could be because an invalid address was passed.
- If the first part of the control block contains non-null data, but the rest consists of zeros, it is again possible that an error occurred while copying data from the users address space, for example, the control block was not placed entirely within valid storage. This could also be due to the control block not being initialized correctly.
- If the error has occurred on exit from MQSeries, it is possible that MQSeries was unable to write the data to the users address space. The data displayed is the version that it was attempting to copy to the users address space.

The control blocks traced

Table 7 illustrates which control blocks are traced for different MQI calls.

Table 7. Control blocks traced for MQSeries MQI calls

MQI call	**Entry**	**Exit**
MQOPEN	MQOD	MQOD
MQCLOSE	None	None
MQPUT	MQMD, MQPMO, and the first 256 bytes of message data	MQMD, MQPMO, and the first 256 bytes of message data
MQPUT1	MQMD, MQOD, MQPMO, and the first 256 bytes of message data	MQMD, MQOD, MQPMO, and the first 256 bytes of message data
MQGET	MQMD, MQGMO	MQMD, MQGMO, and the first 256 bytes of message data
MQINQ	Selectors (if *SelectorCount* is greater than 0)	Selectors (if *SelectorCount* is greater than 0) Integer attributes (if *IntAttrCount* is greater than 0) Character attributes (if *CharAttrLength* is greater than 0)
MQSET	Selectors (if *SelectorCount* is greater than 0) Integer attributes (if *IntAttrCount* is greater than 0) Character attributes (if *CharAttrLength* is greater than 0)	Selectors (if *SelectorCount* is greater than 0) Integer attributes (if *IntAttrCount* is greater than 0) Character attributes (if *CharAttrLength* is greater than 0)

Note: In the special case of an **MQGET** call with the WAIT option, a double entry is seen if there is no message available at the time of the **MQGET** request, but a message subsequently becomes available before the expiry of any time interval specified.

This is because, although the application has issued a single **MQGET** call, the adapter is performing the wait on behalf of the application and when a message becomes available it reissues the call. So in the trace it will appear as a second **MQGET** call.

Information about certain fields of the queue request parameter list is also produced in some circumstances. The fields in this list are identified as follows:

Identifier	**Description**
BufferL	Buffer length
CompCode	Completion code
CharAttL	Character attributes length
DataL	Data length
Hobj	Object handle
IntAttC	Count of integer attributes
pObjDesc	Object descriptor
Options	Options
pBuffer	Address of buffer
pCharAtt	Address of character attributes
pECB	Address of ECB used in get
pGMO	Address of get message options
pIntAtt	Address of integer attributes
pMsgDesc	Address of message descriptor
pPMO	Address of put message options
pSelect	Address of selectors
Reason	Reason code
RSVn	Reserved for IBM
SelectC	Selector count
Thread	Thread
UOWInfo	Information about the unit of work
Userid	CICS or IMS user ID, for batch or TSO this is zero

Examples of trace output

Figure 10 shows an example of a trace taken on entry to an **MQPUT1** call. The following items have been produced:

- Queue request parameter list
- Object descriptor (MQOD)
- Message descriptor (MQMD)
- Put message options (MQPMO)
- The first 256 bytes of message data

Compare this to Figure 11 on page 73 which illustrates the same control blocks on exit from MQSeries.

```
USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQSeries user parameter trace
PUTONE
    Thread... 004C2B10  Userid... CICSUSER  pObjDesc. 106B2010
    pMsgDesc. 106B20B8  pPMO..... 106B2200
    BufferL.. 00000064  pBuffer.. 106A0578  RSV1..... 00000000
    RSV2..... 00000000  RSV3..... 116BC830
    C9E8C1E8  C5C3C9C3  AA8E8583  76270484  | IYAYECIC..ec...d |
    D4D8E3E3  0000048C  00000000  00000000  | MQTT............ |
    00000000  1910C7C2  C9C2D4C9  E8C14BC9  | ......GBIBMIYA.I |
    C7C3E2F2  F0F48E85  83762979  00010000  | GCS204.ec.. .... |

               GMT-01/30/97 14:42:08.412320    LOC-01/30/97 14:42:08.412320

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQSeries user parameter trace
 +0000  D6C44040  00000001  00000000  C2404040  | OD  ........B    |
 +0010  40404040  40404040  40404040  40404040  |                  |
 ...
 +00A0  00000000  00000000                      | ........         |

               GMT-01/30/97 14:42:08.412345    LOC-01/30/97 14:42:08.412345

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQSeries user parameter trace
 +0000  D4C44040  00000001  00000000  00000008  | MD  ............ |
 ...
 +0130  40404040  40404040  40404040  40404040  |                  |
 +0140  40404040                                |                  |

               GMT-01/30/97 14:42:08.412370    LOC-01/30/97 14:42:08.412370

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQSeries user parameter trace
 +0000  D7D4D640  00000001  00000000  FFFFFFFF  | PMO ............ |
 ...
 +0070  40404040  40404040  40404040  40404040  |                  |

               GMT-01/30/97 14:42:08.412393    LOC-01/30/97 14:42:08.412393

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQSeries user parameter trace
 +0000  C1C1C1C1  C1C1C1C1  C1404040  40404040  | AAAAAAAAA        |
 ...
 +0060  40404040                                |                  |

               GMT-01/30/97 14:42:08.412625    LOC-01/30/97 14:42:08.412625
```

Figure 10. Example trace data from an entry trace of an MQPUT1 request

```
USRD9 5EA ASCB 00F87E80           JOBN ECIC330
CSQW073I EXIT:  MQSeries user parameter trace
PUTONE
    Thread... 004C2B10  Userid... CICSUSER  pObjDesc. 106B2010
    pMsgDesc. 106B20B8  pPMO..... 106B2200
    BufferL.. 00000064  pBuffer.. 106A0578  RSV1..... 00000000
    RSV2..... 00000000  RSV3..... 116BC830
    CompCode. 00000002  Reason... 000007FB
    C9E8C1E8  C5C3C9C3  AA8E8583  76270484  | IYAYECIC..ec...d |
    D4D8E3E3  0000048C  00000000  00000000  | MQTT............ |
    00000000  1910C7C2  C9C2D4C9  E8C14BC9  | ......GBIBMIYA.I |
    C7C3E2F2  F0F48E85  83762979  00010000  | GCS204.ec.. .... |
MQRC_OBJECT_TYPE_ERROR

               GMT-01/30/97 14:42:08.412678    LOC-01/30/97 14:42:08.412678

USRD9 5EA ASCB 00F87E80           JOBN ECIC330
CSQW073I EXIT:  MQSeries user parameter trace
 +0000  D6C44040  00000001  00000000  C2404040  | OD  ........B    |
 ...
 +00A0  00000000  00000000                      | ........         |

               GMT-01/30/97 14:42:08.412789    LOC-01/30/97 14:42:08.412789

USRD9 5EA ASCB 00F87E80           JOBN ECIC330
CSQW073I EXIT:  MQSeries user parameter trace
 +0000  D4C44040  00000001  00000000  00000008  | MD  ............ |
 ...
 +0140  40404040                                |                  |

               GMT-01/30/97 14:42:08.412814    LOC-01/30/97 14:42:08.412814

USRD9 5EA ASCB 00F87E80           JOBN ECIC330
CSQW073I EXIT:  MQSeries user parameter trace
 +0000  D7D4D640  00000001  00000000  FFFFFFFF  | PMO ............ |
 ...
 +0070  40404040  40404040  40404040  40404040  |                  |

               GMT-01/30/97 14:42:08.412836    LOC-01/30/97 14:42:08.412836

USRD9 5EA ASCB 00F87E80           JOBN ECIC330
CSQW073I EXIT:  MQSeries user parameter trace
 +0000  C1C1C1C1  C1C1C1C1  C1404040  40404040  | AAAAAAAAA        |
 ...
 +0060  40404040                                |                  |

               GMT-01/30/97 14:42:08.412858    LOC-01/30/97 14:42:08.412858
```

Figure 11. Example trace data from an exit trace of an MQPUT1 request

Other types of trace

You might also find it helpful to use the following trace facilities with MQSeries.

The channel initiator trace

Dumps produced by the channel initiator include a data space called CSQXTRDS containing trace information (see Figure 3 on page 53 for information about how to get a dump of the channel initiator address space). This trace information can be displayed by entering the IPCS command:

```
LIST 1000. DSPNAME(CSQXTRDS)
```

or formatted using the command:

```
CTRACE COMP(CSQXssnm)
```

where *ssnm* is the subsystem name.

The CICS adapter trace

The CICS adapter writes entries to the CICS trace if your trace number is set to a value in the range 0 through 199, and the CICS internal/auxiliary trace is enabled (using the CICS-supplied transaction CETR, or SIT parameters). See the *CICS Problem Determination Guide* manual for information about this.

The trace entries are shown in Appendix C, “CICS adapter trace entries” on page 111.

OS/390 traces

OS/390 traces, which are common to all products operating as formal subsystems of OS/390, are available for use with MQSeries. For information about using and interpreting this trace facility, see the *MVS Diagnosis: Tools and Service Aids* manual.

The IBM service trace

MQSeries collects other, internal, trace information. Sometimes, if an error occurs that you cannot solve yourself, you might be asked by your IBM support center to supply some of this trace data for them to analyze. To do this, use trace identifier 5EE when you format the information as described in “Formatting the information” on page 69.

Part 4. Searching the database for solutions to similar problems

Chapter 10. Searching the database

IBM keeps records of all known problems with its licensed programs on the RETAIN database. IBM support center staff update this database as new problems come to light, and they regularly search the database to see if problems they are told about are already known.

You can use a string of keywords to pinpoint a similar known problem on the software support database. To do this, either use one of the optional search tools such as Information/System or Information/Access, or contact the IBM support center to perform the search for you. If the search is successful, you find a similar problem description and, usually, a fix. If the search is unsuccessful, you should use these keywords when contacting IBM for additional assistance, or when documenting a possible *authorized program analysis report* (APAR).

Before you use the procedures in this section, work through “Preliminary checks” on page 4 to check that the problem does not have a simple solution.

You can use the keyword string (also called the symptom string) that appears in a dump or SYS1.LOGREC record to search the database, or you can build your own keyword string from the procedure described in Chapter 11, “Building a keyword string” on page 81.

Searching the IBM software support database is most effective if you:

- Always spell keywords the way they are spelled in this book
- Include all the appropriate keywords in any discussion with your IBM support center

Search argument process

Use the following procedure when searching the IBM software support database:

1. Search the database using the keywords you have developed. If you have any tools available (such as Information/System), you can search the database yourself. Otherwise, call the IBM support center (see Chapter 12, “IBM program support” on page 97).

   Do *not* use both the CSECT keyword and the load module modifier keyword at the same time for the first search. Refer to “Load module modifier keyword” on page 86 for additional information.
2. Compare each matching APAR closing description with the current failure symptoms.
3. If you find an appropriate APAR, apply the correction or PTF.
4. If you do not find an appropriate APAR, vary the search argument by following the suggestions provided under “Techniques for varying the search” on page 78.
5. If you still cannot find a similar problem, see Chapter 13, “Reporting new problems” on page 101.

Techniques for varying the search

To vary your search, follow these guidelines:

- If you used a complete set of keywords (as described in Chapter 11, “Building a keyword string” on page 81) and were unable to find any problem descriptions to examine, drop one or more of the following keywords and try again:
  - Release-level keyword
  - Load Module modifier keyword
  - Recovery routine modifier keyword
  - CSECT keyword
- If you tried to search with an incomplete set of keywords and found too many problem descriptions to examine, add keywords to narrow your search. For example, for storage manager abends (which produce a reason code beginning with `X'00E2'`), you use the CSECT name recorded in the VRA to narrow or vary the search.
- If you tried to search with a complete set of keywords and found too many matching descriptions *and* if you received a 4-byte MQSeries for OS/390 abend reason code, you might be able to make your set of keywords more precise. Look up the 4-byte abend reason code in the *MQSeries for OS/390 Messages and Codes* manual to find additional information available for this problem.
- If your type-of-failure keyword is WAIT, LOOP, or PERFM, and if you did not find a matching problem description, replace that keyword with one of the other two listed here. Sometimes a problem that appears to be a performance problem might actually be a WAIT or LOOP; likewise, a problem that seems to be a WAIT or a LOOP might actually be recorded as a performance problem.
- If your type-of-failure keyword is MSGx and you received more than one message near the time of the problem, repeat the search replacing the message number in the keyword with the number of each related message in turn.
- If your type-of-failure keyword is MSGx, PERFM, or INCORROUT, and if the problem occurred immediately after you performed some action that an MQSeries book told you to perform, then the problem could be recorded as a DOC type of failure. In this case, try searching with DOC as your type-of-failure keyword, rather than with MSGx, PERFM, or INCORROUT.

The keyword format

The keywords in Chapter 11, “Building a keyword string” on page 81 are described in two distinct formats: the OS/390, or free format; and the structured database (SDB) format. Structured symptoms are also called RETAIN symptoms and “failure keywords”.

If your installation has a tool for performing structured searches, you can use the SDB format. Otherwise, you should use the free format. For both formats, your choice of keywords depends on the type of failure that occurred.

Free format

A free form keyword can consist of any piece of data that is related to the problem. To help you search the data base, a set of keywords has been defined, and you can use them to narrow your search. (For example, if you know the name of the CSECT in error, you can use this to search, but if you add the MSGxx or ABEND keyword, your search will be more precise.)

The following list shows keywords defined for use in a free format search:

Keyword	**Meaning**
ABEND	Abnormal termination of a task; no error message
ABENDxx	Abnormal termination of a task; xx is the abend code
ABENDUxx	User abend; xx is the abend code
DOC	Documentation discrepancy that caused a problem
HALTxx	Halt; xx is the halt number
INCORROUT	Any incorrect data output, except performance degradation
INTEG	Integrity problem
LOOP	Loop
MSGxx	Any message; xx is the message identifier
PERFM	Performance degradation
PROCCHK	Processor check
PROGCH	Program check
WAIT	Wait condition; undocumented and no identifier
WAITxx	System wait condition; xx is the identifier

Structured database (SDB) format

The structured symptoms consist of a prefix keyword, which identifies the type of symptom, followed by a slash (/) and the data portion of the symptom.

- The prefix keyword has one through eight characters.
- All characters must be alphanumeric, #, @, or $.
- At least one character of data is required.
- The maximum length, including the prefix, is 15 characters.

For example, the following is a structured symptom string for a message identifier of CSQC223D:

```
MS/CSQC223D
```

The following list shows the structured symptom strings:

Keyword	**Meaning**
AB	Abend code.
FLDS	Name of a field or control block involved with the problem.
LVLS	Level of the base system or licensed program.
MS	Message identifier.
OPCS	Operation code (opcode) for software, such as an assembler-language opcode.
PCSS	Program command or other software statement, such as JCL, a parameter, or a data set name.
PIDS	Program identifier for a component involved in the problem.
PRCS	Program return code, generated by software, including reason codes and condition codes.
PTFS	Program temporary fix (PTF) for software associated with a problem.
PUBS	Identifier of a publication associated with a problem.
RECS	Record associated with a problem.
REGS	Register for a software program associated with a problem. The value can be the register/PSW difference (*rrddd*), which the STATUS FAILDATA subcommand of IPCS provides for abends. The difference (*ddd*) is a hexadecimal offset from a probable base register or branch register (*rr*).
RIDS	Routine identifier, such as the name of a CSECT or subroutine. If the RIDS/ value has no suffix, the value is a CSECT name. The following suffixes are supported: #L — for a load module #R — for a recovery routine
VALU	Value in a field or register. One of the following qualifiers is required as the first character of the value: B — for a bit value C — for a character value H — for a hexadecimal value
WS	Wait state code issued by the system, or device-issued wait code. One of the following qualifiers is required as the first character of the value: D — for disabled wait (system disabled for I/O or external interrupts) E — for enabled wait

For more information about which prefix keyword to use for which type of symptom, see Appendix A, “SDB format symptom-to-keyword cross-reference” on page 107.

Chapter 11. Building a keyword string

This chapter describes a systematic way of selecting *keywords* to describe a failure in MQSeries for OS/390. Keywords are predefined words or abbreviations that identify aspects of a program failure.

To determine which MQSeries for OS/390 keywords to use and the procedures for selecting them, refer to the flowchart in Figure 12 on page 82.

To begin selecting your keywords:

1. Follow the procedures in “Component-identifier keyword” and “Release-level keyword” on page 83. Do this for all failures.
2. Follow one of the type-of-failure keyword procedures.
3. Identify the area of the failure using CSECT and modifier keywords when appropriate. The procedures in this section refer you to these steps as needed.
4. Follow Chapter 10, “Searching the database” on page 77 to learn how to search the database with your set of keywords. Do this for all failures.
5. If the search is unsuccessful, turn to Chapter 13, “Reporting new problems” on page 101. This helps IBM product support personnel determine whether an APAR should be submitted.

Component-identifier keyword

The *component-identifier keyword* identifies the library within the IBM software support database that contains *authorized program analysis reports* (APARs) and *program temporary fixes* (PTFs) for the product.

The component-identifier keyword for MQSeries for OS/390 is **5655A9500**.

This section describes how to determine the nine-digit component identifier keyword for your failure in order to verify that the problem was caused by MQSeries for OS/390. If the component identifier is not 5655A9500, the problem could be caused by another product.

If the problem caused a dump to be taken, display the dump title, locate the COMP= label, and note the first five characters following that label. If these characters are **A9500**, the problem was caused by MQSeries for OS/390. Append those five characters to **5655** and use this as the first keyword in your search argument.

```
ssnm,ABN=compltn-reason,U=userid,C=compid.release.comp-function,
M=module, LOC=loadmod.csect+csect_offset
```

If you are unable to use the dump title, display the OS/390 SYMPTOM STRING in the formatted dump. Note the nine characters following the PIDS/ label.

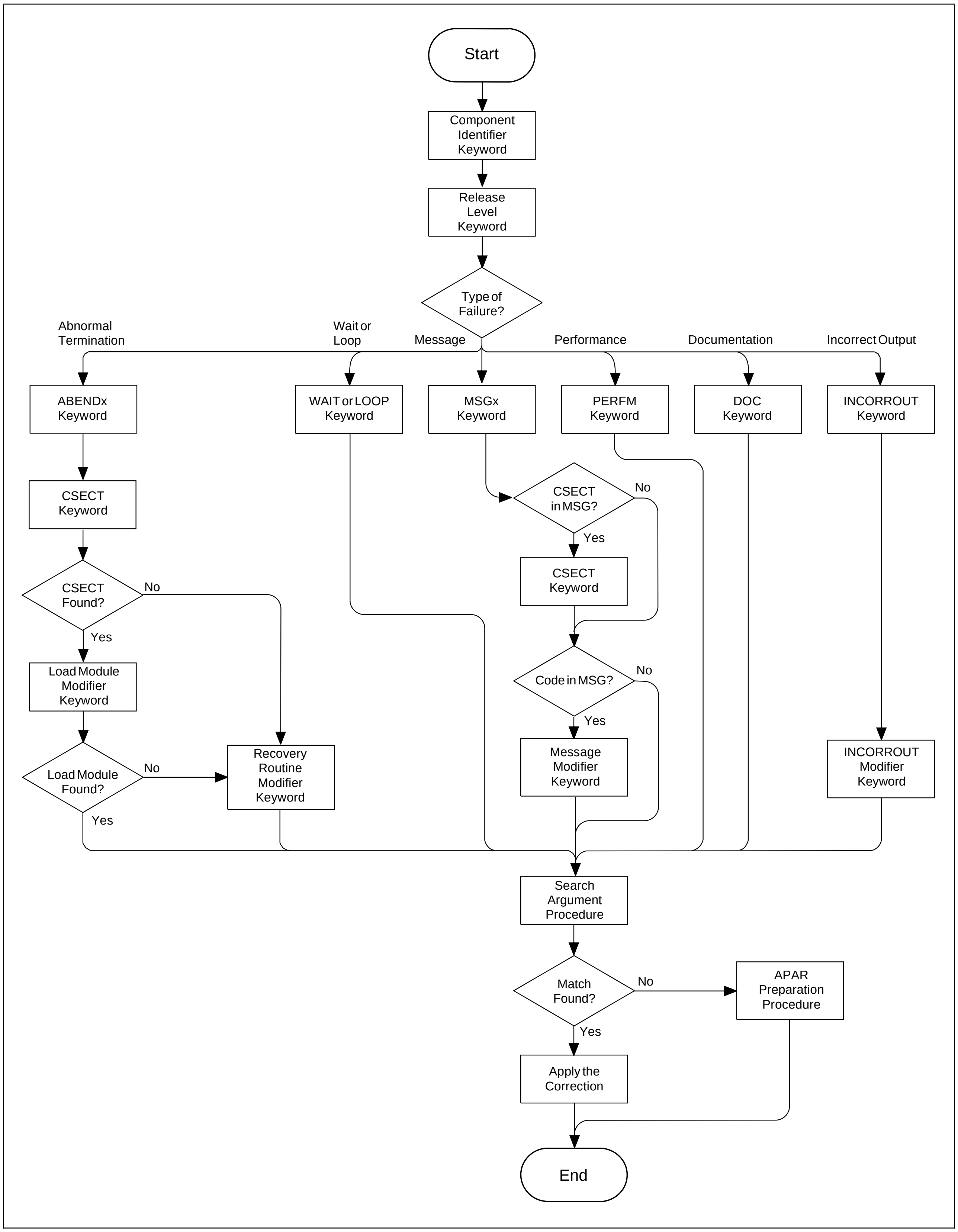


Figure 12. High-level flowchart of various sets of keywords

Release-level keyword

The *release-level keyword* narrows the symptom search to your specific release level. Using this keyword is optional, but recommended, when searching the IBM software support database. It is required, however, when an APAR is submitted.

Locate the three-digit release identifier in the dump title. It follows COMP=A9500, for example:

```
COMP=A9500.210
```

Add this to your keyword string, in one of the formats shown below:

Free format

```
5655A9500 R210
```

Structured format

```
PIDS/5655A9500 LVLS/210
```

Type-of-failure keyword

To narrow your search, use one or more of the type-of-failure and modifier keywords to describe an external symptom of a program failure. The various types of failures are shown in Table 8. Use this table to find the name and page number of the keyword that best matches your problem.

Table 8. Types of MQSeries for OS/390 failures

Problem	Procedure
Abend of the subsystem or task	"Abend keyword" on page 84
Unexpected program suspension	"Wait and loop keywords" on page 88
Uncontrolled program looping (often signaled by repeating messages or output)	"Wait and loop keywords" on page 88
Errors signaled by or associated with messages	"Message keyword" on page 89
Performance degradation	"Performance keyword" on page 91
Documentation problem	"Documentation keyword" on page 92
Unexpected or missing output	"Incorrect output keyword" on page 93

Abend keyword

You should use the ABEND keyword when the subsystem or task terminates abnormally. This procedure describes how to locate the abend completion code and the abend reason code (if there is one), and how to use them in a set of keywords. Check the SYS1.LOGREC to determine how many abends there were. Sometimes an earlier abend causes a secondary abend that causes a dump to be taken. If no dump has been taken, try searching the database with a minimum symptom string (the component-identifier, and release-level keywords). If you cannot find any information that seems to relate to your problem, contact your IBM support center.

When an MQSeries for OS/390 abend occurs, you will see one of the following symptoms:

- An IEA911E message from OS/390, indicating that an SVC dump occurred. See “IEA911E message.”
- The CSQV086E message MQSeries ABNORMAL TERMINATION REASON=xxxxxxxx. See “CSQV086E message” on page 85.

IEA911E message

1. Use the `DISPLAY DUMP,TITLE` command on the console to display the SVC dump title for this abend, or use one of the methods described in Chapter 8, “MQSeries dumps” on page 51 to look at the dump title in the dump.

   Note: If the first five characters of the COMP field are not A9500, or the dump title is not of the same form as Figure 5 on page 62 or Figure 6 on page 63, the problem was not caused by MQSeries for OS/390, or you are looking at the wrong dump.

2. Locate the 3-character completion code following the word ABND.

   - If the completion code is `X'071'`, or `X'122'`, the operator pressed the RESTART key or canceled the job, probably to break a loop. Verify that this is the case, and turn to “Wait and loop keywords” on page 88.
   - Otherwise, add this to your keyword string, in one of the formats shown below (in this example, `X'0C4'` is used):

     Free format

     ```
     5655A9500 R210 ABEND0C4
     ```

     Structured format

     ```
     PIDS/5655A9500 LVLS/210 AB/S00C4
     ```

3. Some abends also have reason codes. These reason codes are usually found in message CSQV086E, and register 15 at the time of the abend. Locate the reason code for the abend either:

   - In the 4-byte reason code field in a dump title generated by MQSeries for OS/390
   - In the registers at time of error in the abstract information section of the dump
   - From the value of register 15 in the error summary display

4. If the completion code is X'5C6', review the diagnostic information for the reason code in the *MQSeries for OS/390 Messages and Codes* manual. Follow any procedures recommended there.

   If the completion code is anything else, and you have found a reason code, check the value against the description of the abend code in the *MVS System Codes* manual to see if it is valid for the abend completion code.

5. Add the reason code to the keyword string (in this example X'00E20015' is used):

   Free format

   ```
   5655A9500 R210 ABEND5C6 RC00E20015
   ```

   Structured format

   ```
   PIDS/5655A9500 LVLS/210 AB/S05C6 PRCS/00E20015
   ```

   Turn to “CSECT keyword” on page 86.

CSQV086E message

1. Issue the DISPLAY DUMP command to see whether any SVC dumps occurred near the time the message appeared. (See the *MVS System Commands* manual if necessary.)
2. If there was only one SVC dump for the abend, follow the procedure starting at step 1 on page 84.
3. If there were two or more SVC dumps, follow the steps below.
   a. Read the sections in the *MQSeries for OS/390 Messages and Codes* manual that describe the reason code appearing in your message, and any reason codes appearing in the SVC dump titles. Reason codes appear after the completion code in the SVC dump title. For an example, see “Analyzing the dump” on page 62.
   b. Compare the reason codes in the SVC dumps to determine which dump relates to the CSQV086E message.
   c. Use that SVC dump and follow the procedure starting at step 1 on page 84.
4. If there were two or more *different* abends, follow the steps below:
   a. Determine which abend was the original cause by reviewing the time stamps in the SYS1.LOGREC entries.
   b. Use that SVC dump and follow the procedure starting at step 1 on page 84.
5. If there were no SVC dumps for the abend, follow the steps below.
   a. Locate the 4-byte reason code in the message.
   b. Review the diagnostic information in the *MQSeries for OS/390 Messages and Codes* manual. Follow any procedures recommended there.
   c. Add this to your keyword string, in one of the formats shown below (in this example, a reason code of X'00D93001' is used):

      Free format

      ```
      5655A9500 R210 ABEND6C6 RC00D93001
      ```

      Structured format

      ```
      PIDS/5655A9500 LVLS/210 AB/S06C6 PRCS/00D93001
      ```

Turn to "CSECT keyword."

CSECT keyword

To find the name of the failing CSECT, locate the LOC= label; the second word following it is the CSECT name. For an example, see "Analyzing the dump" on page 62.

Any CSECT name you locate should begin with the letters CSQ. If you find a CSECT name with a different prefix, the problem is probably not in MQSeries for OS/390.

Add the CSECT name to your keyword string:

Free format

```
5655A9500 R210 ABEND0C4 CSQVATRM
```

Structured format

```
PIDS/5655A9500 LVLS/210 AB/S00C4  RIDS/CSQVATRM
```

If required, narrow your search further by referring to "Load module modifier keyword."

If you cannot find the CSECT, turn to "Recovery routine modifier keyword" on page 87.

Load module modifier keyword

Use the load module modifier keyword to identify the name of the load module involved if your search using the CSECT keyword was unsuccessful, or yielded too many possible matches:

- If your search was unsuccessful, replace the CSECT name with the load module name and try again.
- If your search yielded too many possible matches, add the load module name to your string to further narrow the search.

All MQSeries for OS/390 load module names begin with CSQ. If you follow these instructions and find a load module name with a different prefix, the problem is in another product.

To locate the load module name, locate the first word following the label LOC=. This is the load module name, and it precedes the CSECT name. (For an example, see "Analyzing the dump" on page 62.)

Add the load module name to your keyword string, or substitute it for the CSECT name as appropriate. If you are using the structured format, follow the name of the module with the characters #L to indicate that this is a load module. Search the database again using the revised keyword string. (See Chapter 10, "Searching the database" on page 77.)

Free format

```
5655A9500 R210 ABEND5C6 RC00E50013 CSQSLD1 CSQSVSTK
```

(with load module name and then CSECT name)

```
5655A9500 R210 ABEND5C6 RC00E50013 CSQSLD1
```

(with load module name only)

Structured format

`PIDS/5655A9500 LVLS/210 AB/S05C6 PRCS/00E50013` **`RIDS/CSQSLD1#L`** `RIDS/CSQSVSTK`
(with load module name and then CSECT name)

`PIDS/5655A9500 LVLS/210 AB/S05C6 PRCS/00E50013` **`RIDS/CSQSLD1#L`**
(with load module name only)

Recovery routine modifier keyword

Include the name of the recovery routine only when you were unable to determine the names of the CSECT and load module involved at the time of failure, after looking in both the SVC dump and the SYS1.LOGREC entry.

To obtain the recovery routine name, locate the area of the dump title containing the symbol M=. The word following this identifies the functional recovery routine (FRR) or the extended specify task abnormal exit (ESTAE). For an example, see “Analyzing the dump” on page 62.

Add this word to your keyword string. If you are using the structured format, follow the name of the module with the characters #R to indicate that this is a recovery routine. Search the database (see Chapter 10, “Searching the database” on page 77).

Free format

`5655A9500 R210 ABEND5C6 RC00E20015` **`CSQTFRCV`**

Structured format

`PIDS/5655A9500 LVLS/210 AB/S05C6 PRCS/00E20015` **`RIDS/CSQTFRCV#R`**

Wait and loop keywords

If the problem occurred immediately after you did something an MQSeries manual told you to do, the problem might be related to the manual. If you think that this is the case, turn to “Documentation keyword” on page 92.

If you have verified that the wait or loop problem cannot be resolved through other means, use the following procedure:

1. Add WAIT or LOOP to your keyword string, in one of the formats shown below (in this example **WAIT** is used).

   Free format

   ```
   5655A9500 R210 WAIT
   ```

   Structured format

   ```
   PIDS/5655A9500 LVLS/210 WAIT
   ```

2. Turn to Chapter 10, “Searching the database” on page 77.

Message keyword

Use the MSG keyword if an error is associated with an MQSeries for OS/390 message. If you received multiple messages for one error, search the database using the first message issued. If unsuccessful, search the database using the next message, then the next, and so on.

To see if other messages related to your problem have been issued, check the console for MQSeries for OS/390 messages, as well as messages issued by other products. If any message is prefixed with "IEC", indicating it was issued by data management services, check the SYSLOG for messages that identify associated data set problems. SYSLOG can also help to diagnose user errors.

If your message was issued immediately after you did something that an MQSeries manual told you to do, the problem might be related to the documentation rather than to the message. If this is the case, turn to "Documentation keyword" on page 92. Otherwise, compare the message prefix with those shown in the table below to determine the appropriate procedure to follow.

Table 9 (Page 1 of 2). Message prefixes

Prefix	Component	Procedure
AMQ	MQSeries (not OS/390)	Consult *MQSeries Messages*
ATB	APPC	Consult *MVS System Messages*
ATR	Resource recovery services	Consult *MVS System Messages*
CBC	C/C++	Consult *C/MVS User's Guide*
CEE	Language Environment®	Consult *Language Environment Debugging Guide and Run-Time Messages*
CSQ	MQSeries for OS/390	Follow "Procedure for MQSeries for OS/390 messages" on page 90
CSV	Contents supervision	Consult *MVS System Messages*
DFH	CICS	Consult *CICS Messages and Codes*
DFS	IMS	Consult *IMS/ESA Messages and Codes*
DSN	DB2®	Consult *DB2 Messages and Codes*
EDC	Language Environment	Consult *Language Environment Debugging Guide and Run-Time Messages*
EZA, EZB, EZY	TCP/IP	Consult *TCP/IP for MVS Messages and Codes*
IBM	Language Environment	Consult *Language Environment Debugging Guide and Run-Time Messages*
ICH	RACF®	Consult *OS/390 Security Server (RACF) Messages and Codes*
IDC	Access method services	Consult *MVS System Messages*
IEA	OS/390 system services	Consult *MVS System Messages*
IEC	Data management services	Consult *MVS System Messages*
IEE, IEF	OS/390 system services	Consult *MVS System Messages*
IKJ	TSO	Consult *MVS System Messages*

Table 9 (Page 2 of 2). Message prefixes

Prefix	Component	Procedure
IST	VTAM	Consult *VTAM Messages*
IWM	MVS workload management services	Consult *MVS System Messages*
IXC	Cross-system coupling facility	Consult *MVS System Messages*

Procedure for MQSeries for OS/390 messages

1. Check whether the name of the CSECT issuing the message appears. This name follows the message number. If no CSECT name appears, only one CSECT can issue this message.
2. Determine whether the message contains any variables, such as return or reason codes.
3. If no CSECT name appears, add the message number to your keyword string, in one of the formats shown below (in this example, message CSQJ006I is used):

   Free format
   ```
   5655A9500 R210 MSGCSQJ006I
   ```
   Structured format
   ```
   PIDS/5655A9500 LVLS/210 MS/CSQJ006I
   ```
4. If a CSECT name does appear, add both the message number and the CSECT name to your keyword string, in one of the formats shown below (in this example, a message number of CSQJ311E and a CSECT name of CSQJC005 are used):

   Free format
   ```
   5655A9500 R210 MSGCSQJ311E CSQJC005
   ```
   Structured format
   ```
   PIDS/5655A9500 LVLS/210 MS/CSQJ311E RIDS/CSQJC005
   ```
5. If the message contains return or reason codes, add these to your keyword string, in one of the formats shown below:

   Free format
   ```
   5655A9500 R210 MSGCSQM002I RCE
   ```
   Structured format
   ```
   PIDS/5655A9500 LVLS/210 MS/CSQM002I PRCS/0000000E
   ```
6. If the message contains any other types of variables, append them to your keyword string.

   Free format
   ```
   5655A9500 R210 MSGCSQJ104I OPEN
   ```
   Structured format
   ```
   PIDS/5655A9500 LVLS/210 MS/CSQJ1041 MS/OPEN
   ```
7. Turn to Chapter 10, “Searching the database” on page 77.

Performance keyword

Most performance problems can be resolved through system tuning and should be handled by the MQSeries for OS/390 system administrator. Before following the procedure below, use this checklist to verify that the performance problem cannot be resolved through other means:

- Refer to Chapter 5, “Dealing with performance problems” on page 29 to see if the way you have designed your MQSeries for OS/390 subsystem and applications can be changed to improve their performance.
- Verify that the performance problem is not related to a WAIT or LOOP. Turn to Chapter 4, “Dealing with waits and loops” on page 23.
- If the problem occurred immediately after you did something an MQSeries manual told you to do, the problem might be related to the manual. Turn to “Documentation keyword” on page 92.
- If performance degraded after someone tuned MQSeries for OS/390, verify that the tuning options selected were appropriate. Perhaps the problem can be resolved by choosing other options.

If you have verified that the performance problem cannot be resolved through other means, use the following procedure:

1. Record the actual performance, expected performance, and source of expected performance criteria.
2. Add PERFM to your keyword string, as shown below, and turn to Chapter 10, “Searching the database” on page 77.

   Free format
   ```
   5655A9500 R210 PERFM
   ```

   Structured format
   ```
   PIDS/5655A9500 LVLS/210 PERFM
   ```
3. If required, you can narrow your search by adding free-format keywords that describe what you were doing when you experienced the performance problem.

Documentation keyword

The DOC keyword identifies problems caused by incorrect or missing information in an MQSeries manual. It is possible that a documentation problem could be detected when trying to resolve problems with messages, incorrect output, and performance.

Use the following procedure if you need to use the DOC keyword in your keyword string:

1. Locate the incomplete or erroneous information. Note the page, or topic number, and describe the error and the resulting problem.
2. Add the document number, hyphens omitted, to your keyword string, in one of the formats shown below (in this example, the document number for this manual (GC34-5376-00) is used):

   Free format
   ```
   5655A9500 R210 DOC GC34537600
   ```

   Structured format
   ```
   PIDS/5655A9500 LVLS/210 PUBS/GC34537600
   ```

   Turn to Chapter 10, “Searching the database” on page 77.

   If your search is unsuccessful, follow Step 3.
3. Broaden your search by replacing the last two digits with two asterisks (**). This searches for all problems on that document, rather than on a specific release of the document.

   Free format
   ```
   5655A9500 R210 DOC GC345376**
   ```

   Structured format
   ```
   PIDS/5655A9500 LVLS/210 PUBS/GC345376**
   ```

   If your search is unsuccessful, follow Step 4.
4. If the problem is severe, consider initiating a DOC APAR. Use the information gathered in Step 1, and turn to Chapter 13, “Reporting new problems” on page 101.
5. If the problem is less severe, include your suggestions on the Readers’ Comment Form in the back of that manual, or send it electronically as described at the back of the manual. Include your name and address if you want a reply.

   Corrections resulting from readers’ comments are included in future editions of the manual but are not included in the software support database.

Incorrect output keyword

Use the INCORROUT keyword when output was expected but not received, or when output was different from expected. However, if this problem occurred after you did something that an MQSeries manual told you to do, the manual could be in error. If this is the case, refer to “Documentation keyword” on page 92.

1. Add **INCORROUT** to your existing keyword string.

   Free format
   ```
   5655A9500 R210 INCORROUT
   ```
   Structured format
   ```
   PIDS/5655A9500 LVLS/210 INCORROUT
   ```
2. Determine the function and secondary modifier keywords for your problem from Table 10.
3. Add the modifier keywords to your string and use it to search the database. See Chapter 10, “Searching the database” on page 77.

   Free format
   ```
   5655A9500 R210 INCORROUT RECOVERY BACKOUT
   ```
   Structured format
   ```
   PIDS/5655A9500 LVLS/210 INCORROUT RECOVERY BACKOUT
   ```

Table 10. INCORROUT modifier keywords

Function keyword	Secondary keywords	Problem occurrence
RECOVERY	none	During recovery
	BACKOUT	At backout time
	CHECKPOINT	At checkpoint time
	COMMIT	At commit time
	LOGGING	During logging
	RECOVER	During attempt to recover in-doubt
	RESTART	During restart process
UTILITY	none	While running a utility
	CSQ1LOGP	While running CSQ1LOGP
	CHANGE LOG	While using the Change Log Inventory utility
	PRINT LOGMAP	While using the Print Log Map utility
	COMMAND	While running the COMMAND function
	COPY	While running the COPY function
	EMPTY	While running the EMPTY function
	FORMAT	While running the FORMAT function
	LOAD	While running the LOAD function

Part 5. Working with IBM to solve your problem

Chapter 12. IBM program support

The IBM Customer Engineering Program Support structure exists to help you resolve problems with IBM products, and to ensure that you can make the best use of your IBM computing systems. Program support is available to all licensed users of IBM licensed programs; you can get assistance by contacting your local support center.

This chapter helps you to decide when to contact the support center, and what information you need to have collected before doing so. The chapter also gives you an understanding of the way in which IBM Program Support works.

When to contact the support center

Before contacting the support center, try to ensure that the problem belongs with the center. Don't worry if you can't be sure that the problem is due to MQSeries for OS/390 itself. How sure you are depends on the complexity of your installation, the experience and skill levels of your systems staff, and the symptoms that your system has been showing.

In practice, quite a lot of errors reported to Program Support turn out to be user errors, or they cannot be reproduced, or they need to be dealt with by other parts of IBM Service such as Hardware CE or Systems Engineering. This indicates just how difficult it can be to determine the precise cause of a problem. User errors are mainly caused by bugs in application programs, and mistakes in setting up systems.

Dealing with the support center

Your first contact at the support center is the call receipt operator, who takes initial details and puts your problem on a queue. You are subsequently contacted by a support center representative, and your problem is taken from there.

The support center needs to know as much as possible about your problem, and you should have the information ready before making your first call. It is a good idea to write the information on a problem reporting sheet such as the one shown in Figure 13 on page 98.

There are two advantages of using a problem reporting sheet when contacting the IBM support center:

- You are communicating with the IBM support center by telephone. With all your findings before you on a sheet of paper, you are likely to be better prepared to respond to the questions that you may be asked.
- You should maintain your own in-house tracking system for problems. A problem tracking system records and documents all problems. This information can then be used for planning, organizing, communication, and establishing priorities for controlling and resolving these problems.

```
PROBLEM REPORTING SHEET

Date                     Severity                  Problem  No.

                                                   Incident No.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Problem/Enquiry

Abend                    Incorrout                 OS/390 Rel.

Wait                     Module                    MQSeries Rel.

Loop                     Message                   CICS Rel.

Performance              Other                     IMS Rel.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Documentation available

Abend                    System dump               Program output

Message                  Transaction dump          Other

Trace                    Translator output

Symptom string           Compiler output

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Actions

Date          Name              Activity








- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Resolution

APAR                     PTF                       Other
```

Figure 13. Sample problem reporting sheet

What the support center needs to know

When you contact the support center, you need to give the operator the name of your organization and your *access code*. Your access code is a unique code authorizing you to use IBM Software Services, and you provide it every time you contact the center. Using this information, the operator consults your customer profile, which contains information about your address, relevant contact names, telephone numbers, and details of the IBM products at your installation.

The support center operator asks whether this is a new problem, or a further call on an existing one. If it is new, it is assigned a unique *incident number*. A *problem management record* (PMR) is opened on the RETAIN system, where all activity associated with your problem is recorded. The problem remains ‘open’ until resolved.

Make a note of the incident number on your own problem reporting sheet. The support center expects you to quote the incident number in all future calls connected with this problem.

If the problem is new to you, the operator asks you for the source of the problem within your system software—that is, the program that seems to be the cause of the problem. Because you are reading this manual, it is likely that you have already identified MQSeries for OS/390 as the problem source. You also have to give the version and release number.

You need to give the *severity level* for the problem. Severity levels can be 1, 2, or 3, and they have the following meanings:

Level 1 This indicates that you are unable to use the system, and have a critical condition that needs immediate attention.

Level 2 This indicates that you are able to use the system, but that operation is severely restricted.

Level 3 This indicates that you are able to use the program, with limited functions, but the problem is not critical to your overall operation.

When deciding the severity of the problem, take care neither to understate it, nor to overstate it. The support center procedures depend on the severity level so that the most appropriate use can be made of the center’s skills and resources. A severity level 1 problem is normally dealt with immediately.

Finally, the call receipt operator asks you for a brief description of the problem, and might prompt you to quote the MQSeries for OS/390 symptom string, or to give any keywords associated with the problem. The primary keywords are ABEND, WAIT, LOOP, PERFM, INCORROUT, MSG, and DOC, corresponding exactly with the problem classification types used in Chapter 11, “Building a keyword string” on page 81. Strings containing other keywords are also useful. These are not predefined, and might include such items as a message or message number, an abend code, any parameters known to be associated with the problem, or, for example, STARTUP or INITIALIZATION.

The keywords are subsequently used as search arguments on the RETAIN database, to see if your problem is a known one that has already been the subject of an *authorized program analysis report* (APAR).

You are not asked for any more information at this stage. However, you need to keep all the information relevant to the problem, and any available documentation such as dumps, traces, and translator, compiler, and program output.

How your problem is subsequently progressed depends on its nature. The representative who handles the problem gives you guidance on what is required from you. The possibilities are described in the next section.

What happens next

Details of your call are passed to the appropriate support group using the RETAIN problem management system. Your problem, assuming it is one associated with MQSeries for OS/390, is put on the MQSeries for OS/390 queue. The problems are dealt with in order of receipt and severity level.

At first, an IBM support center representative uses the keywords that you have provided to search the RETAIN database. If your problem is found to be one already known to IBM, and a fix has been devised for it, a *program temporary fix* (PTF) can be dispatched to you quickly.

If the RETAIN search is unsuccessful, you are asked to provide more information about your problem.

Let the representative know if any of the following events occurred before the problem appeared:

- Changes in the level of OS/390 or licensed programs
- Regenerations
- PTFs applied
- Additional features used
- Application programs changed
- Unusual operator action

You might be asked to give values from a formatted dump or trace table, or to carry out some special activity, for example to set a trap, or to use trace with a certain type of selectivity, and then to report the results.

It might be necessary to have several follow-up telephone calls, depending on the complexity of the symptoms and your system environment. In every case, the actions taken by you and the support center are entered in the PMR. The representative can then be acquainted with the full history of the problem before any follow-up call.

The result of the investigations determines whether your problem is a new one, or one that is already known. If it is already known, and a fix has been developed, the fix is sent to you.

If the problem is new, an APAR can be submitted. This is dealt with by the MQSeries for OS/390 *change team*. What you need to do is described in Chapter 13, “Reporting new problems” on page 101.

Chapter 13. Reporting new problems

An *authorized program analysis report* (APAR) is your means of informing the appropriate change team of a problem you have found with an IBM program.

When the change team solves the problem, they might produce a local fix enabling you to get your system running properly again. Finally, a *program temporary fix* (PTF) is produced to replace the module in error, and the APAR is closed.

The APAR process

The first step in the APAR process is that an IBM support center representative enters your APAR into the RETAIN system. The APAR text contains a description of your problem. If you have found a means of getting round the problem, details of this are entered as well. Your name is also entered, so that the support center know whom to contact if the change team needs to ask anything further about the APAR documentation.

When the APAR has been entered, you are given an APAR number. You must write this number on all the documentation you submit to the change team. This number is always associated with the APAR and its resolution and, if a code change is required, with the fix as well.

The next stage in the APAR process—getting the relevant documentation to the change team—is up to you.

Here is a summary of the things you need to do:

1. You must collect all of the documentation that is required for the APAR. You are given guidance by the IBM support center representative on precisely what you need to send. The documentation that is required varies, depending on the problem area, but “Collecting the documentation for the APAR” gives you an idea of the material that you are asked to supply.
2. You need to package all the documentation and send it to the change team. The procedure for this is given in “Sending the documentation to the change team” on page 102.
3. The change team might ask you to test the fix on your system.
4. Lastly, you need to apply the PTF resulting from the APAR when it becomes available.

Collecting the documentation for the APAR

As a general rule, the documentation you need to submit for an APAR includes all the material you need yourself to do problem determination. Some of the documentation is common to all MQSeries for OS/390 problems, and some is specific to particular types of problem.

Make sure the problem you have described can be seen in the documentation you send. If the problem has ambiguous symptoms, you need to reveal the sequence of events leading up to the failure. Tracing is valuable in this respect but you might be able to provide details that trace cannot give. You are encouraged to annotate your documentation, if your annotation is legible and if it does not cover up vital information. You can highlight data in any hardcopy you send, using transparent highlighting markers. You can also write notes in the margins, preferably using a red pen so that the notes are not overlooked.

Finally, note that if you send too little documentation, or if it is unreadable, the change team will have to return the APAR marked “insufficient documentation”. It is, therefore, worthwhile preparing your documentation carefully and sending everything relevant to the problem.

The general documentation is described below. However, these are only guidelines—you must find out from the IBM support center representative precisely what documentation you need to send for your specific problem.

General documentation needed for all problems with MQSeries for OS/390

Here is a list of the general documentation you might be asked to submit for an APAR:

- Any hardcopy or softcopy illustrating the symptoms of the problem
- The dump of the problem
- The appropriate SYS1.LOGREC records
- The console log
- A portion of the MQSeries for OS/390 recovery log
- Trace records
- Trace information produced by the CICS or IMS adapter
- Buffer pool statistics
- Listings of relevant application programs
- A list of PTFs and APARs applied

Because of the size of SVC dumps in the cross memory environment, transfer the SYS1.DUMPxx data set to a tape or like device. You can use the PRDMP service aid program to transfer the SYS1.DUMPxx data set contents to another data set for archiving until the problem is resolved. Depending on the nature of the problem, the IBM support center might ask you to send the entire dump on tape. This allows the support center to extract any additional data needed for problem resolution; for example, CSA, SQA, or the private storage area.

Sending the documentation to the change team

The documentation you submit for the problem is best shipped in an APAR box, which you can obtain from your local IBM branch. APAR boxes are clearly marked as such, and they have a special panel where tracking information such as the APAR number can be written.

Each item submitted must have the following information attached and visible:

- The APAR number assigned by IBM
- A list of data sets on the tape (application source program, JCL, or data)
- A description of how the tape was made, including:
  - The exact JCL listing or the list of commands used
  - The recording mode and density
  - Tape labeling

– The record format, logical record length, and block size used for each data set

Packing and mailing the APAR box

Place all your documentation and notes in one or more APAR boxes, making sure that the boxes are marked, for example, “1 of 2”, and so on, if you need to use more than one.

If you include any magnetic tapes, write this clearly on the outside of the box.

Note: Some countries employ local methods of shipping APARs to change teams. Please check with your IBM support center for details.

When the change team receives the package, this is noted on your APAR record on the RETAIN system. The team then investigates the problem. Sometimes, they will have to ask you for more documentation, perhaps specifying some trap that you must apply to get it.

When the problem is solved, a code is entered on RETAIN to close the APAR, and you are provided with a fix.

You can enquire any time at your support center on how your APAR is progressing, particularly if it is a problem of high severity.

Applying the fix

When the change team have found a fix for your problem, they might want you to test it on your system. If they ask you to test the fix, you are normally given two weeks to do it and to provide them with the results. However, you can ask for an extension if you are unable to complete the testing in that time.

When the team is confident that the fix is satisfactory, the APAR is certified by the MQSeries for OS/390 development team and the APAR is closed. You receive notification when this happens.

The APAR becomes a PTF

If the solution involves a change to code in a module that you can assemble, you are sent the code change right away. The change is later distributed as a PTF.

If you are unable to assemble it yourself, because it involves a part of MQSeries for OS/390 that is object serviced, you might be supplied with a ZAP or a TOTEST PTF.

If you want a PTF to resolve a specific problem, you can order it explicitly by its PTF number through the IBM support center. Otherwise, you can wait for the PTF to be sent out on the standard distribution tape.

Part 6. Appendixes

Appendix A. SDB format symptom-to-keyword cross-reference

Table 11 (Page 1 of 2). SDB format symptom-to-keyword cross-reference

Symptom	Keyword		Symptom	Keyword
abend	AB/		access method	RIDS/
address	ADRS/		APAR	PTFS/
assembler macro	RIDS/		assembler message	MS/
CLIST	RIDS/		command	PCSS/
compiler message	MS/		completion code	PRCS/
component	PIDS/		condition code	PRCS/
control block	FLDS/		control block offset	ADRS/
control register	REGS/		CSECT	RIDS/
data set name	PCSS/		dependent component	PIDS/
device error code	PRCS/		disabled wait (coded)	WS/
displacement	ADRS/		display	DEVS/
document	PUBS/		DSECT	FLDS/
enabled wait (coded)	WS/		error code	PRCS/
EXEC	RIDS/		feedback code	PRCS/
field	FLDS/		field value	VALU/
file mode	PCSS/		file name	PCSS/
file type	PCSS/		flag	FLDS/
floating-point register	REGS/		full-screen mode	PCSS/
function key	PCSS/		general purpose register	REGS/
hang	WS/		hung user or task	WS/
I/O operator codes	OPCS/		incorrect output	INCORROUT*
JCL card	PCSS/		JCL parameter	PCSS/
job step code	PRCS/		key	PCSS/
label, code	FLDS/		language statement	PCSS/
level	LVLS/		library name	PCSS/
line command	PCSS/		loop	LOOP*
low core address	ADRS/		machine check	SIG/
macro as a routine	RIDS/		macro as a statement	PCSS/
maintenance level	PTFS/		message	MS/
module	RIDS/		offset	ADRS/
opcode	OPCS/		operator command	PCSS/
operator key	PCSS/		operator message	MS/
option	PCSS/		overlay	OVS/
PA key	PCSS/		panel	RIDS/

Table 11 (Page 2 of 2). SDB format symptom-to-keyword cross-reference

Symptom	Keyword		Symptom	Keyword
parameter	PCSS/		performance	PERFM*
PF key	PCSS/		procedure name	PCSS/
process name	PCSS/		profile option	PCSS/
program check	AB/		program id	RIDS/
program key	PCSS/		program statement	PCSS/
PSW	FLDS/		PTF, PE or otherwise	PTFS/
publication	PUBS/		PUT level	PTFS/
reason code	PRCS/		register value	VALU/
register	REGS/		release level	LVLS/
reply to message	PCSS/		reply to prompt	PCSS/
request code	OPCS/		response to message	PCSS/
response to prompt	PCSS/		return code	PRCS/
routine	RIDS/		service level	PTFS/
special character	PCSS/		SRL	PUBS/
statement	PCSS/		status code	PRCS/
step code	PRCS/		structure word	FLDS/
subroutines	RIDS/		SVC	OPCS/
SYSGEN parameter	PCSS/		system check	PRCS/
table	FLDS/		terminal key	PCSS/
value	VALU/		variable	FLDS/
wait (coded)	WS/		wait (uncoded)	WAIT*

Note: An asterisk (*) indicates that there is no prefix keyword for this type of problem. Use the type-of-failure keyword shown for searches of the software support database.

Appendix B. MQSeries component and resource manager identifiers

Table 12 (Page 1 of 2). MQSeries component and resource manager identifiers

ID	Prefix	Hex ID	Component name	RMID
—	CMQ	—	Application header files	—
m	CSQm	X'94'	Connection manager	148
A	CSQA	X'C1'	Application interface	—
B	CSQB	X'C2'	Batch adapter	—
C	CSQC	X'C3'	CICS adapter	—
F	CSQF	X'C6'	Message generator	24
G	CSQG	X'C7'	Functional recovery manager	199
H	CSQH	X'C8'	Security manager interface	200
I	CSQI	X'C9'	Data manager	201
J	CSQJ	X'D1'	Recovery log manager	4
K	CSQK	X'D2'	Distributed queuing (using CICS ISC)	—
L	CSQL	X'D3'	Lock manager	211
M	CSQM	X'D4'	Message manager	212
N	CSQN	X'D5'	Command server	213
O	CSQO	X'D6'	Operations and control	—
P	CSQP	X'D7'	Buffer manager	215
Q	CSQQ	X'D8'	IMS adapter	—
R	CSQR	X'D9'	Recovery manager	3
S	CSQS	X'E2'	Storage manager	6
T	CSQT	X'E3'	Timer services	227
U	CSQU	X'E4'	Utilities	—
V	CSQV	X'E5'	Agent services	2
W	CSQW	X'E6'	Instrumentation facilities	16, 26
X	CSQX	X'E7'	Distributed queuing	231
Y	CSQY	X'E8'	Initialization procedures and general services	1
Z	CSQZ	X'E9'	System parameter manager	12
1	CSQ1	X'F1'	Service facilities	—
2	CSQ2	X'F2'	MQSeries-IMS bridge	242
3	CSQ3	X'F3'	Subsystem support	7, 8
4	CSQ4	X'F4'	Sample programs	—
6	CSQ6	X'F6'	Customization	—
7	CSQ7	X'F7'	Dump formatting	—
8	CSQ8	X'F8'	Installation	—
9	CSQ9	X'F9'	Generalized command preprocessor	23

Table 12 (Page 2 of 2). MQSeries component and resource manager identifiers

ID	Prefix	Hex ID	Component name	RMID
—	DMQ	—	Internet gateway	—
—	IMQ	—	C++ bindings	—

Appendix C. CICS adapter trace entries

The CICS trace entry for these values is AP0*xxx* (where *xxx* is the trace number you specified when the CICS adapter was enabled). These trace entries are all issued by CSQCTRUE, except CSQCTEST which is issued by CSQCRST and CSQCDSP.

Table 13 (Page 1 of 2). CICS adapter trace entries

Name	Description	Trace sequence	Trace data
CSQCABNT	Abnormal termination	Before issuing END_THREAD ABNORMAL to MQSeries. This is due to the end of the task and therefore an implicit backout could be performed by the application. A ROLLBACK request is included in the END_THREAD call in this case.	Unit of work information. You can use this information when finding out about the status of work. (For example, it can be verified against the output produced by the DISPLAY THREAD command, or the log print utility.)
CSQCBACK	Syncpoint backout	Before issuing BACKOUT to MQSeries. This is due to an explicit backout request from the application.	Unit of work information.
CSQCCCRC	Completion code and reason code	After unsuccessful return from API call.	Completion code and reason code.
CSQCCOMM	Syncpoint commit	Before issuing COMMIT to MQSeries. This can be due to a single-phase commit request or the second phase of a two-phase commit request. The request is due to a explicit syncpoint request from the application.	Unit of work information.
CSQCDCFF	IBM use only		
CSQCEXER	Execute resolve	Before issuing EXECUTE_RESOLVE to MQSeries.	The unit of work information of the unit of work issuing the EXECUTE_RESOLVE. This is the last in-doubt unit of work in the resynchronization process.
CSQCGETW	GET wait	Before issuing CICS wait.	Address of the ECB to be waited on.
CSQCGMGD	GET message data	After successful return from **MQGET**.	Up to 40 bytes of the message data.
CSQCGMGH	GET message handle	Before issuing **MQGET** to MQSeries.	Object handle.
CSQCGMGI	Get message ID	After successful return from **MQGET**.	Message ID and correlation ID of the message.
CSQCINDL	In-doubt list	After successful return from the second INQUIRE_INDOUBT.	The in-doubt units of work list.
CSQCINDO	IBM use only		
CSQCINDS	In-doubt list size	After successful return from the first INQUIRE_INDOUBT and the in-doubt list is not empty.	Length of the list; divided by 64 gives the number of in-doubt units of work.
CSQCINDW	Syncpoint in doubt	During syncpoint processing, CICS is in doubt as to the disposition of the unit of work.	Unit of work information.
CSQCINQH	INQ handle	Before issuing **MQINQ** to MQSeries.	Object handle.
CSQCLOSH	CLOSE handle	Before issuing **MQCLOSE** to MQSeries.	Object handle.

Table 13 (Page 2 of 2). CICS adapter trace entries

Name	Description	Trace sequence	Trace data
CSQCLOST	Disposition lost	During the resynchronization process, CICS informs the adapter that it has been cold started so no disposition information regarding the unit of work being resynchronized is available.	Unit of work ID known to CICS for the unit of work being resynchronized.
CSQCNIND	Disposition not in doubt	During the resynchronization process, CICS informs the adapter that the unit of work being resynchronized should not have been in doubt (that is, perhaps it is still running).	Unit of work ID known to CICS for the unit of work being resynchronized.
CSQCNORT	Normal termination	Before issuing END_THREAD NORMAL to MQSeries. This is due to the end of the task and therefore an implicit syncpoint commit might be performed by the application. A COMMIT request is included in the END_THREAD call in this case.	Unit of work information.
CSQCOPNH	OPEN handle	After successful return from **MQOPEN**.	Object handle.
CSQCOPNO	OPEN object	Before issuing **MQOPEN** to MQSeries.	Object name.
CSQCPMGD	PUT message data	Before issuing **MQPUT** to MQSeries.	Up to 40 bytes of the message data.
CSQCPMGH	PUT message handle	Before issuing **MQPUT** to MQSeries.	Object handle.
CSQCPMGI	PUT message ID	After successful **MQPUT** from MQSeries.	Message ID and correlation ID of the message.
CSQCPREP	Syncpoint prepare	Before issuing PREPARE to MQSeries in the first phase of two-phase commit processing. This call can also be issued from the distributed queuing component as an API call.	Unit of work information.
CSQCP1MD	PUTONE message data	Before issuing **MQPUT1** to MQSeries.	Up to 40 bytes of data of the message.
CSQCP1MI	PUTONE message ID	After successful return from **MQPUT1**.	Message ID and correlation ID of the message.
CSQCP1ON	PUTONE object name	Before issuing **MQPUT1** to MQSeries.	Object name.
CSQCRBAK	Resolved backout	Before issuing RESOLVE_ROLLBACK to MQSeries.	Unit of work information.
CSQCRCMT	Resolved commit	Before issuing RESOLVE_COMMIT to MQSeries.	Unit of work information.
CSQCRMIR	RMI response	Before returning to the CICS RMI (resource manager interface) from a specific invocation.	Architected RMI response value. Its meaning depends of the type of the invocation. To determine the type of invocation, look at previous trace entries produced by the CICS RMI component.
CSQCRSYN	Resync	Before the resynchronization process starts for the task.	Unit of work ID known to CICS for the unit of work being resynchronized.
CSQCSETH	SET handle	Before issuing **MQSET** to MQSeries.	Object handle.
CSQCTASE	IBM use only		
CSQCTEST	Trace test	Used in EXEC CICS ENTER TRACE call to verify the trace number supplied by the user or the trace status of the connection.	No data.

Appendix D. Examples of CEDF output

This appendix gives examples of the output produced by the CICS execution diagnostic facility (CEDF) when using MQSeries. The examples show the data produced on entry to and exit from the following MQI calls, in both hexadecimal and character format:

- “MQOPEN”
- “MQCLOSE” on page 114
- “MQPUT” on page 115
- “MQPUT1” on page 116
- “MQGET” on page 118
- “MQINQ” on page 119
- “MQSET” on page 121

MQOPEN

The parameters for this call are:

ARG 000	Connection handle
ARG 001	Object descriptor
ARG 002	Options
ARG 003	Object handle
ARG 004	Completion code
ARG 005	Reason code

```
  STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')             AT X'05ECAFD8'
 001: ARG 001 (X'D6C44040000000010000000 1C3C5C4C6')             AT X'00144910'
 001: ARG 002 (X'00000072000000000000000000000000')             AT X'001445E8'
 001: ARG 003 (X'00000000000000720000000000000000')             AT X'001445E4'
 001: ARG 004 (X'00000000000000000000000000000000')             AT X'001445EC'
 001: ARG 005 (X'00000000000000000000000000000000')             AT X'001445F0'
```

Figure 14. Example CEDF output on entry to an MQOPEN call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')             AT X'05ECAFD8'
 001: ARG 001 (X'D6C4404000000001000000 01C3C5C4C6')             AT X'00144910'
 001: ARG 002 (X'00000072000000000000000000000000')             AT X'001445E8'
 001: ARG 003 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 004 (X'00000000000000000000000000000000')             AT X'001445EC'
 001: ARG 005 (X'00000000000000000000000000000000')             AT X'001445F0'
```

Figure 15. Example CEDF output on exit from an MQOPEN call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('.............. .')
 001: ARG 001 ('OD  ........CEDF')
 001: ARG 002 ('................')
 001: ARG 003 ('................')
 001: ARG 004 ('................')
 001: ARG 005 ('................')
```

Figure 16. Example CEDF output on entry to an MQOPEN call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('.............. .')
 001: ARG 001 ('OD  ........CEDF')
 001: ARG 002 ('................')
 001: ARG 003 ('................')
 001: ARG 004 ('................')
 001: ARG 005 ('................')
```

Figure 17. Example CEDF output on exit from an MQOPEN call (character)

MQCLOSE

The parameters for this call are:

ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Options
ARG 003	Completion code
ARG 004	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')            AT X'001445E4'
 001: ARG 002 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 003 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 004 (X'00000000000000080000000800000060')            AT X'001445F0'
```

Figure 18. Example CEDF output on entry to an MQCLOSE call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000000000007200000000')            AT X'001445E0'
 001: ARG 001 (X'00000000000000720000000000000000')            AT X'001445E4'
 001: ARG 002 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 003 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 004 (X'00000000000000080000000800000060')            AT X'001445F0'
```

Figure 19. Example CEDF output on exit from an MQCLOSE call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('.............. .')
 001: ARG 003 ('................')
 001: ARG 004 ('...............-')
```

Figure 20. Example CEDF output on entry to an MQCLOSE call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('.............. .')
 001: ARG 003 ('................')
 001: ARG 004 ('...............-')
```

Figure 21. Example CEDF output on exit from an MQCLOSE call (character)

MQPUT

The parameters for this call are:

ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Message descriptor
ARG 003	Put message options
ARG 004	Buffer length
ARG 005	Message data
ARG 006	Completion code
ARG 007	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')            AT X'001445E4'
 001: ARG 002 (X'D4C44040000000010000000000000008')            AT X'001449B8'
 001: ARG 003 (X'D7D4D640000000010000002400000000')            AT X'00144B48'
 001: ARG 004 (X'00000008000000000000000000040000')            AT X'001445F4'
 001: ARG 005 (X'D4D8E285998985A20000000000000000')            AT X'00144BF8'
 001: ARG 006 (X'00000000000000000000000800000000')            AT X'001445EC'
 001: ARG 007 (X'00000000000000080000000000000000')            AT X'001445F0'
```

Figure 22. Example CEDF output on entry to an MQPUT call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')             AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 002 (X'D4C44040000000010000000000000008')             AT X'001449B8'
 001: ARG 003 (X'D7D4D640000000010000002400000000')             AT X'00144B48'
 001: ARG 004 (X'00000008000000000000000000040000')             AT X'001445F4'
 001: ARG 005 (X'D4D8E285998985A20000000000000000')             AT X'00144BF8'
 001: ARG 006 (X'00000000000000000000000800000000')             AT X'001445EC'
 001: ARG 007 (X'00000000000000080000000000000000')             AT X'001445F0'
```

Figure 23. Example CEDF output on exit from an MQPUT call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO ............')
 001: ARG 004 ('................')
 001: ARG 005 ('MQSeries........')
 001: ARG 006 ('................')
 001: ARG 007 ('................')
```

Figure 24. Example CEDF output on entry to an MQPUT call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO ............')
 001: ARG 004 ('................')
 001: ARG 005 ('MQSeries........')
 001: ARG 006 ('................')
 001: ARG 007 ('................')
```

Figure 25. Example CEDF output on exit from an MQPUT call (character)

MQPUT1

The parameters for this call are:

ARG 000	Connection handle
ARG 001	Object descriptor
ARG 002	Message descriptor
ARG 003	Put message options
ARG 004	Buffer length
ARG 005	Message data
ARG 006	Completion code
ARG 007	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000000000007200000000')             AT X'001445E0'
 001: ARG 001 (X'D6C44040000000100000001C3C5C4C6')              AT X'00144910'
 001: ARG 002 (X'D4C44040000000010000000000000008')             AT X'001449B8'
 001: ARG 003 (X'D7D4D640000000010000002400000000')             AT X'00144B48'
 001: ARG 004 (X'00000008000000080000006000040000')             AT X'001445F4'
 001: ARG 005 (X'D4D8E285998985A2D4D8E285998985A2')             AT X'00144BF8'
 001: ARG 006 (X'00000000000000000000000800000008')             AT X'001445EC'
 001: ARG 007 (X'00000000000000080000000800000060')             AT X'001445F0'
```

Figure 26. Example CEDF output on entry to an MQPUT1 call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000000000007200000000')             AT X'001445E0'
 001: ARG 001 (X'D6C44040000000100000001C3C5C4C6')              AT X'00144910'
 001: ARG 002 (X'D4C44040000000010000000000000008')             AT X'001449B8'
 001: ARG 003 (X'D7D4D640000000010000002400000000')             AT X'00144B48'
 001: ARG 004 (X'00000008000000080000006000040000')             AT X'001445F4'
 001: ARG 005 (X'D4D8E285998985A2D4D8E285998985A2')             AT X'00144BF8'
 001: ARG 006 (X'00000000000000000000000800000008')             AT X'001445EC'
 001: ARG 007 (X'00000000000000080000000800000060')             AT X'001445F0'
```

Figure 27. Example CEDF output on exit from an MQPUT1 call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('OD  ........CEDF')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO ............')
 001: ARG 004 ('...........-....')
 001: ARG 005 ('MQSeriesMQSeries')
 001: ARG 006 ('................')
 001: ARG 007 ('...............-')
```

Figure 28. Example CEDF output on entry to an MQPUT1 call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('OD  ........CEDF')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO ............')
 001: ARG 004 ('...........-....')
 001: ARG 005 ('MQSeriesMQSeries')
 001: ARG 006 ('................')
 001: ARG 007 ('...............-')
```

Figure 29. Example CEDF output on exit from an MQPUT1 call (character)

MQGET

The parameters for this call are:

ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Message descriptor
ARG 003	Get message options
ARG 004	Buffer length
ARG 005	Message buffer
ARG 006	Message length
ARG 007	Completion code
ARG 008	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')             AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 002 (X'D4C44040000000010000000000000000')             AT X'001449B8'
 001: ARG 003 (X'C7D4D6400000000100004044FFFFFFFF')             AT X'00144B00'
 001: ARG 004 (X'00000080000000000000000000040000')             AT X'001445F4'
 001: ARG 005 (X'00000000000000000000000000000000')             AT X'00144C00'
 001: ARG 006 (X'00000000000000000004000000000000')             AT X'001445F8'
 001: ARG 007 (X'00000000000000000000008000000000')             AT X'001445EC'
 001: ARG 008 (X'00000000000000800000000000000000')             AT X'001445F0'
```

Figure 30. Example CEDF output on entry to an MQGET call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')             AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 002 (X'D4C44040000000010000000000000008')             AT X'001449B8'
 001: ARG 003 (X'C7D4D6400000000100004044FFFFFFFF')             AT X'00144B00'
 001: ARG 004 (X'00000080000000080000000000040000')             AT X'001445F4'
 001: ARG 005 (X'D4D8E285998985A20000000000000000')             AT X'00144C00'
 001: ARG 006 (X'00000008000000000004000000000000')             AT X'001445F8'
 001: ARG 007 (X'00000000000000000000008000000008')             AT X'001445EC'
 001: ARG 008 (X'00000000000000800000000800000000')             AT X'001445F0'
```

Figure 31. Example CEDF output on exit from an MQGET call (hexadecimal)

```
 STATUS:  ABOUT TO EXECUTE COMMAND
 CALL TO RESOURCE MANAGER MQM
  001: ARG 000 ('................')
  001: ARG 001 ('................')
  001: ARG 002 ('MD  ............')
  001: ARG 003 ('GMO ...... .....')
  001: ARG 004 ('................')
  001: ARG 005 ('................')
  001: ARG 006 ('................')
  001: ARG 007 ('................')
  001: ARG 008 ('................')
```

Figure 32. Example CEDF output on entry to an MQGET call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('GMO ...... .....')
 001: ARG 004 ('................')
 001: ARG 005 ('MQSeries........')
 001: ARG 006 ('................')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
```

Figure 33. Example CEDF output on exit from an MQGET call (character)

MQINQ

The parameters for this call are:

ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Count of selectors
ARG 003	Array of attribute selectors
ARG 004	Count of integer attributes
ARG 005	Integer attributes
ARG 006	Length of character attributes buffer
ARG 007	Character attributes
ARG 008	Completion code
ARG 009	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')             AT X'05ECAFCC'
 001: ARG 001 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 002 (X'000000020000404485ECA00885ECA220')             AT X'05ECAFD4'
 001: ARG 003 (X'0000000D0000000C0000000000000000')             AT X'00144C08'
 001: ARG 004 (X'000000020000404485ECA00885ECA220')             AT X'05ECAFD4'
 001: ARG 005 (X'00000000000000000000000000000000')             AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')             AT X'05ECAFCC'
 001: ARG 007 (X'00000000000000000000000000000000')             AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')             AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800040000')             AT X'001445F0'
```

Figure 34. Example CEDF output on entry to an MQINQ call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')             AT X'05ECAFCC'
 001: ARG 001 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 002 (X'000000020000404485ECA00885ECA220')             AT X'05ECAFD4'
 001: ARG 003 (X'0000000D0000000C0040000000000000')             AT X'00144C08'
 001: ARG 004 (X'000000020000404485ECA00885ECA220')             AT X'05ECAFD4'
 001: ARG 005 (X'00400000000000000000000000000000')             AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')             AT X'05ECAFCC'
 001: ARG 007 (X'00000000000000000000000000000000')             AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')             AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800040000')             AT X'001445F0'
```

Figure 35. Example CEDF output on exit from an MQINQ call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('.............. .')
 001: ARG 001 ('................')
 001: ARG 002 ('...... .e...e.s.')
 001: ARG 003 ('................')
 001: ARG 004 ('...... .e...e.s.')
 001: ARG 005 ('................')
 001: ARG 006 ('.............. .')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
 001: ARG 009 ('................')
```

Figure 36. Example CEDF output on entry to an MQINQ call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('.............. .')
 001: ARG 001 ('................')
 001: ARG 002 ('...... .e...e.s.')
 001: ARG 003 ('......... ......')
 001: ARG 004 ('...... .e...e.s.')
 001: ARG 005 ('. ..............')
 001: ARG 006 ('.............. .')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
 001: ARG 009 ('................')
```

Figure 37. Example CEDF output on exit from an MQINQ call (character)

MQSET

The parameters for this call are:

ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Count of selectors
ARG 003	Array of attribute selectors
ARG 004	Count of integer attributes
ARG 005	Integer attributes
ARG 006	Length of character attributes buffer
ARG 007	Character attributes
ARG 008	Completion code
ARG 009	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')             AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 002 (X'00000001000000020000404485ECA008')             AT X'05ECAFDC'
 001: ARG 003 (X'00000018000007DF0000000000000000')             AT X'00144C08'
 001: ARG 004 (X'00000001000000020000404485ECA008')             AT X'05ECAFDC'
 001: ARG 005 (X'00000000000000000000000000000000')             AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')             AT X'05ECAFD8'
 001: ARG 007 (X'00000000000000000000000000000000')             AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')             AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800000060')             AT X'001445F0'
```

Figure 38. Example CEDF output on entry to an MQSET call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')             AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')             AT X'001445E4'
 001: ARG 002 (X'000000010000000200004044485ECA008')            AT X'05ECAFDC'
 001: ARG 003 (X'00000018000007DF0000000000000000')             AT X'00144C08'
 001: ARG 004 (X'000000010000000200004044485ECA008')            AT X'05ECAFDC'
 001: ARG 005 (X'00000000000000000000000000000000')             AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')             AT X'05ECAFD8'
 001: ARG 007 (X'00000000000000000000000000000000')             AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')             AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800000060')             AT X'001445F0'
```

Figure 39. Example CEDF output on exit from an MQSET call (hexadecimal)

```
 STATUS:  ABOUT TO EXECUTE COMMAND
 CALL TO RESOURCE MANAGER MQM
  001: ARG 000 ('................')
  001: ARG 001 ('................')
  001: ARG 002 ('.......... .e...')
  001: ARG 003 ('................')
  001: ARG 004 ('.......... .e...')
  001: ARG 005 ('................')
  001: ARG 006 ('.............. .')
  001: ARG 007 ('................')
  001: ARG 008 ('................')
  001: ARG 009 ('...............-')
```

Figure 40. Example CEDF output on entry to an MQSET call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('.......... .e...')
 001: ARG 003 ('................')
 001: ARG 004 ('.......... .e...')
 001: ARG 005 ('................')
 001: ARG 006 ('.............. .')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
 001: ARG 009 ('...............-')
```

Figure 41. Example CEDF output on exit from an MQSET call (character)

Appendix E. Notices

Programming interface information

This book is intended to help you diagnose problems with MQSeries for OS/390. This book documents information that is Diagnosis, Modification, or Tuning Information provided by MQSeries for OS/390.

Attention: Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

Trademarks

Part 7. Glossary and index

Glossary of terms and abbreviations

This glossary defines MQSeries terms and abbreviations used in this book. If you do not find the term you are looking for, see the Index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

A

abend reason code. A 4-byte hexadecimal code that uniquely identifies a problem with MQSeries for OS/390. A complete list of MQSeries for OS/390 abend reason codes and their explanations is contained in the *MQSeries for OS/390 Messages and Codes* manual.

active log. See *recovery log*.

adapter. An interface between MQSeries for OS/390 and TSO, IMS, CICS, or batch address spaces. An adapter is an attachment facility that enables applications to access MQSeries services.

address space. The area of virtual storage available for a particular job.

address space identifier (ASID). A unique, system-assigned identifier for an address space.

administrator commands. MQSeries commands used to manage MQSeries objects, such as queues, processes, and namelists.

alert. A message sent to a management services focal point in a network to identify a problem or an impending problem.

alert monitor. In MQSeries for OS/390, a component of the CICS adapter that handles unscheduled events occurring as a result of connection requests to MQSeries for OS/390.

alias queue object. An MQSeries object, the name of which is an alias for a base queue defined to the local queue manager. When an application or a queue manager uses an alias queue, the alias name is resolved and the requested operation is performed on the associated base queue.

allied address space. See *ally*.

ally. An OS/390 address space that is connected to MQSeries for OS/390.

alternate user security. A security feature in which the authority of one user ID can be used by another user ID; for example, to open an MQSeries object.

APAR. Authorized program analysis report.

application environment. The software facilities that are accessible by an application program. On the OS/390 platform, CICS and IMS are examples of application environments.

application queue. A queue used by an application.

archive log. See *recovery log*.

ARM. Automatic Restart Management

ASID. Address space identifier.

asynchronous messaging. A method of communication between programs in which programs place messages on message queues. With asynchronous messaging, the sending program proceeds with its own processing without waiting for a reply to its message. Contrast with *synchronous messaging*.

attribute. One of a set of properties that defines the characteristics of an MQSeries object.

authorization checks. Security checks that are performed when a user tries to issue administration commands against an object, for example to open a queue or connect to a queue manager.

authorized program analysis report (APAR). A report of a problem caused by a suspected defect in a current, unaltered release of a program.

Automatic Restart Management (ARM). An OS/390 recovery function that can improve the availability of specific batch jobs or started tasks, and therefore result in faster resumption of productive work.

B

backout. An operation that reverses all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins. Contrast with *commit*.

basic mapping support (BMS). An interface between

CICS and application programs that formats input and output display data and routes multiple-page output messages without regard for control characters used by various terminals.

BMS. Basic mapping support.

bootstrap data set (BSDS). A VSAM data set that contains:

- An inventory of all active and archived log data sets known to MQSeries for OS/390
- A wrap-around inventory of all recent MQSeries for OS/390 activity

The BSDS is required if the MQSeries for OS/390 subsystem has to be restarted.

browse. In message queuing, to use the MQGET call to copy a message without removing it from the queue. See also *get*.

browse cursor. In message queuing, an indicator used when browsing a queue to identify the message that is next in sequence.

BSDS. Bootstrap data set.

buffer pool. An area of main storage used for MQSeries for OS/390 queues, messages, and object definitions. See also *page set*.

C

call back. In MQSeries, a requester message channel initiates a transfer from a sender channel by first calling the sender, then closing down and awaiting a call back.

CCF. Channel control function.

CCSID. Coded character set identifier.

CDF. Channel definition file.

channel. See *message channel*.

channel control function (CCF). In MQSeries, a program to move messages from a transmission queue to a communication link, and from a communication link to a local queue, together with an operator panel interface to allow the setup and control of channels.

channel definition file (CDF). In MQSeries, a file containing communication channel definitions that associate transmission queues with communication links.

channel event. An event indicating that a channel instance has become available or unavailable. Channel events are generated on the queue managers at both ends of the channel.

checkpoint. A time when significant information is written on the log. Contrast with *syncpoint*.

CI. Control interval.

CL. Control Language.

client. A run-time component that provides access to queuing services on a server for local user applications. The queues used by the applications reside on the server. See also *MQSeries client*.

client application. An application, running on a workstation and linked to a client, that gives the application access to queuing services on a server.

client connection channel type. The type of MQI channel definition associated with an MQSeries client. See also *server connection channel type*.

cluster. A network of queue managers that are logically associated in some way.

cluster queue. A queue that is hosted by a cluster queue manager and made available to other queue managers in the cluster.

cluster queue manager. A queue manager that is a member of a cluster. A queue manager may be a member of more than one cluster.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

command. In MQSeries, an administration instruction that can be carried out by the queue manager.

command prefix (CPF). In MQSeries for OS/390, a character string that identifies the queue manager to which MQSeries for OS/390 commands are directed, and from which MQSeries for OS/390 operator messages are received.

command processor. The MQSeries component that processes commands.

command server. The MQSeries component that reads commands from the system-command input queue, verifies them, and passes valid commands to the command processor.

commit. An operation that applies all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins. Contrast with *backout*.

completion code. A return code indicating how an MQI call has ended.

connect. To provide a queue manager connection handle, which an application uses on subsequent MQI calls. The connection is made either by the MQCONN call, or automatically by the MQOPEN call.

connection handle. The identifier or token by which a program accesses the queue manager to which it is connected.

context. Information about the origin of a message.

context security. In MQSeries, a method of allowing security to be handled such that messages are obliged to carry details of their origins in the message descriptor.

control interval (CI). A fixed-length area of direct access storage in which VSAM stores records and creates distributed free spaces. The control interval is the unit of information that VSAM transmits to or from direct access storage.

controlled shutdown. See *quiesced shutdown*.

CPF. Command prefix.

D

DAE. Dump analysis and elimination.

datagram. The simplest message that MQSeries supports. This type of message does not require a reply.

DCI. Data conversion interface.

dead-letter queue (DLQ). A queue to which a queue manager or application sends messages that it cannot deliver to their correct destination.

default object. A definition of an object (for example, a queue) with all attributes defined. If a user defines an object but does not specify all possible attributes for that object, the queue manager uses default attributes in place of any that were not specified.

deferred connection. A pending event that is activated when a CICS subsystem tries to connect to MQSeries for OS/390 before MQSeries for OS/390 has been started.

distributed application. In message queuing, a set of application programs that can each be connected to a different queue manager, but that collectively constitute a single application.

distributed queue management (DQM). In message queuing, the setup and control of message channels to queue managers on other systems.

DLQ. Dead-letter queue.

DQM. Distributed queue management.

dual logging. A method of recording MQSeries for OS/390 activity, where each change is recorded on two data sets, so that if a restart is necessary and one data set is unreadable, the other can be used. Contrast with *single logging*.

dual mode. See *dual logging*.

dump analysis and elimination (DAE). An OS/390 service that enables an installation to suppress SVC dumps and ABEND SYSUDUMP dumps that are not needed because they duplicate previously written dumps.

dynamic queue. A local queue created when a program opens a model queue object. See also *permanent dynamic queue* and *temporary dynamic queue*.

E

environment. See *application environment*.

ESM. External security manager.

ESTAE. Extended specify task abnormal exit.

event. See *channel event*, *instrumentation event*, *performance event*, and *queue manager event*.

event data. In an event message, the part of the message data that contains information about the event (such as the queue manager name, and the application that gave rise to the event). See also *event header*.

event header. In an event message, the part of the message data that identifies the event type of the reason code for the event.

event message. Contains information (such as the category of event, the name of the application that caused the event, and queue manager statistics) relating to the origin of an instrumentation event in a network of MQSeries systems.

event queue. The queue onto which the queue manager puts an event message after it detects an event. Each category of event (queue manager, performance, or channel event) has its own event queue.

extended specify task abnormal exit (ESTAE). An OS/390 macro that provides recovery capability and gives control to the specified exit routine for processing, diagnosing an abend, or specifying a retry address.

external security manager (ESM). A security product that is invoked by the OS/390 System Authorization Facility. RACF is an example of an ESM.

F

FIFO. First-in-first-out.

first-in-first-out (FIFO). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time. (A)

forced shutdown. A type of shutdown of the CICS adapter where the adapter immediately disconnects from MQSeries for OS/390, regardless of the state of any currently active tasks. Contrast with *quiesced shutdown*.

FRR. Functional recovery routine.

functional recovery routine (FRR). An OS/390 recovery/termination manager facility that enables a recovery routine to gain control in the event of a program interrupt.

G

GCPC. Generalized command preprocessor.

generalized command preprocessor (GCPC). An MQSeries for OS/390 component that processes MQSeries commands and runs them.

Generalized Trace Facility (GTF). An OS/390 service program that records significant system events, such as supervisor calls and start I/O operations, for the purpose of problem determination.

get. In message queuing, to use the MQGET call to remove a message from a queue. See also *browse*.

global trace. An MQSeries for OS/390 trace option where the trace data comes from the entire MQSeries for OS/390 subsystem.

GTF. Generalized Trace Facility.

H

handle. See *connection handle* and *object handle*.

I

IFCID. A trace event number.

immediate shutdown. In MQSeries, a shutdown of a queue manager that does not wait for applications to disconnect. Current MQI calls are allowed to complete, but new MQI calls fail after an immediate shutdown has been requested. Contrast with *quiesced shutdown* and *preemptive shutdown*.

in-doubt unit of recovery. In MQSeries, the status of a unit of recovery for which a syncpoint has been requested but not yet confirmed.

initialization input data sets. Data sets used by MQSeries for OS/390 when it starts up.

initiation queue. A local queue on which the queue manager puts trigger messages.

input/output parameter. A parameter of an MQI call in which you supply information when you make the call, and in which the queue manager changes the information when the call completes or fails.

input parameter. A parameter of an MQI call in which you supply information when you make the call.

instrumentation event. A facility that can be used to monitor the operation of queue managers in a network of MQSeries systems. MQSeries provides instrumentation events for monitoring queue manager resource definitions, performance conditions, and channel conditions. Instrumentation events can be used by a user-written reporting mechanism in an administration application that displays the events to a system operator. They also allow applications acting as agents for other administration networks to monitor reports and create the appropriate alerts.

Interactive Problem Control System (IPCS). A component of OS/390 that permits online problem management, interactive problem diagnosis, online debugging for disk-resident abend dumps, problem tracking, and problem reporting.

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialog manager. It is used for writing application programs, and provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

IPCS. Interactive Problem Control System.

ISPF. Interactive System Productivity Facility.

L

listener. In MQSeries distributed queuing, a program that monitors for incoming network connections.

local definition. An MQSeries object belonging to a local queue manager.

local definition of a remote queue. An MQSeries object belonging to a local queue manager. This object defines the attributes of a queue that is owned by another queue manager. In addition, it is used for queue-manager aliasing and reply-to-queue aliasing.

local queue. A queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with *remote queue*.

local queue manager. The queue manager to which a program is connected and that provides message queuing services to the program. Queue managers to which a program is not connected are called *remote queue managers*, even if they are running on the same system as the program.

log. In MQSeries, a file recording the work done by queue managers while they receive, transmit, and deliver messages, to enable them to recover in the event of failure.

logical unit of work (LUW). See *unit of work*.

M

machine check interrupt. An interruption that occurs as a result of an equipment malfunction or error. A machine check interrupt can be either hardware recoverable, software recoverable, or nonrecoverable.

MCA. Message channel agent.

MCI. Message channel interface.

message. (1) In message queuing applications, a communication sent between programs. See also *persistent message* and *nonpersistent message*. (2) In system programming, information intended for the terminal operator or system administrator.

message channel. In distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender at one end and a receiver at the other end) and a communication link. Contrast with *MQI channel*.

message channel agent (MCA). A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue. See also *message queue interface*.

message channel interface (MCI). The MQSeries interface to which customer- or vendor-written programs that transmit messages between an MQSeries queue manager and another messaging system must conform. A part of the MQSeries Framework.

message descriptor. Control information describing the message format and presentation that is carried as part of an MQSeries message. The format of the message descriptor is defined by the MQMD structure.

message priority. In MQSeries, an attribute of a message that can affect the order in which messages on a queue are retrieved, and whether a trigger event is generated.

message queue. Synonym for *queue*.

message queue interface (MQI). The programming interface provided by the MQSeries queue managers. This programming interface allows application programs to access message queuing services.

message queuing. A programming technique in which each program within an application communicates with the other programs by putting messages on queues.

message sequence numbering. A programming technique in which messages are given unique numbers during transmission over a communication link. This enables the receiving process to check whether all messages are received, to place them in a queue in the original order, and to discard duplicate messages.

messaging. See *synchronous messaging* and *asynchronous messaging*.

model queue object. A set of queue attributes that act as a template when a program creates a dynamic queue.

MQI. Message queue interface.

MQI channel. Connects an MQSeries client to a queue manager on a server system, and transfers only MQI calls and responses in a bidirectional manner. Contrast with *message channel*.

MQSC. MQSeries commands.

MQSeries. A family of IBM licensed programs that provides message queuing services.

N

namelist. An MQSeries object that contains a list of names, for example, queue names.

nonpersistent message. A message that does not survive a restart of the queue manager. Contrast with *persistent message*.

null character. The character that is represented by `X'00'`.

O

object. In MQSeries, an object is a queue manager, a queue, a process definition, a channel, a namelist, or a storage class (OS/390 only).

object descriptor. A data structure that identifies a particular MQSeries object. Included in the descriptor are the name of the object and the object type.

object handle. The identifier or token by which a program accesses the MQSeries object with which it is working.

off-loading. In MQSeries for OS/390, an automatic process whereby a queue manager's active log is transferred to its archive log.

output log-buffer. In MQSeries for OS/390, a buffer that holds recovery log records before they are written to the archive log.

output parameter. A parameter of an MQI call in which the queue manager returns information when the call completes or fails.

P

page set. A VSAM data set used when MQSeries for OS/390 moves data (for example, queues and messages) from buffers in main storage to permanent backing storage (DASD).

pending event. An unscheduled event that occurs as a result of a connect request from a CICS adapter.

percolation. In error recovery, the passing along a preestablished path of control from a recovery routine to a higher-level recovery routine.

performance event. A category of event indicating that a limit condition has occurred.

performance trace. An MQSeries trace option where the trace data is to be used for performance analysis and tuning.

permanent dynamic queue. A dynamic queue that is deleted when it is closed only if deletion is explicitly requested. Permanent dynamic queues are recovered if the queue manager fails, so they can contain persistent messages. Contrast with *temporary dynamic queue*.

persistent message. A message that survives a restart of the queue manager. Contrast with *nonpersistent message*.

ping. In distributed queuing, a diagnostic aid that uses the exchange of a test message to confirm that a message channel or a TCP/IP connection is functioning.

platform. In MQSeries, the operating system under which a queue manager is running.

point of recovery. In MQSeries for OS/390, the term used to describe a set of backup copies of MQSeries for OS/390 page sets and the corresponding log data sets required to recover these page sets. These backup copies provide a potential restart point in the event of page set loss (for example, page set I/O error).

preemptive shutdown. In MQSeries, a shutdown of a queue manager that does not wait for connected applications to disconnect, nor for current MQI calls to complete. Contrast with *immediate shutdown* and *quiesced shutdown*.

process definition object. An MQSeries object that contains the definition of an MQSeries application. For example, a queue manager uses the definition when it works with trigger messages.

program temporary fix (PTF). A solution or by-pass of a problem diagnosed by IBM field engineering as the result of a defect in a current, unaltered release of a program.

PTF. Program temporary fix.

Q

queue. An MQSeries object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages—they point to other queues, or can be used as models for dynamic queues.

queue manager. (1) A system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. See also *local queue manager* and *remote queue manager*. (2) An MQSeries object that defines the attributes of a particular queue manager.

queue manager event. An event that indicates:

- An error condition has occurred in relation to the resources used by a queue manager. For example, a queue is unavailable.
- A significant change has occurred in the queue manager. For example, a queue manager has stopped or started.

queuing. See *message queuing*.

quiesced shutdown. (1) In MQSeries, a shutdown of a queue manager that allows all connected applications to disconnect. Contrast with *immediate shutdown* and *preemptive shutdown*. (2) A type of shutdown of the CICS adapter where the adapter disconnects from MQSeries, but only after all the currently active tasks have been completed. Contrast with *forced shutdown*.

quiescing. In MQSeries, the state of a queue manager prior to it being stopped. In this state, programs are allowed to finish processing, but no new programs are allowed to start.

R

RBA. Relative byte address.

reason code. A return code that describes the reason for the failure or partial success of an MQI call.

receiver channel. In message queuing, a channel that responds to a sender channel, takes messages from a communication link, and puts them on a local queue.

recovery log. In MQSeries for OS/390, data sets containing information needed to recover messages, queues, and the MQSeries subsystem. MQSeries for OS/390 writes each record to a data set called the *active log*. When the active log is full, its contents are off-loaded to a DASD or tape data set called the *archive log*. Synonymous with *log*.

recovery termination manager (RTM). A program that handles all normal and abnormal termination of tasks by passing control to a recovery routine associated with the terminating function.

relative byte address (RBA). The displacement in bytes of a stored record or control interval from the beginning of the storage space allocated to the data set to which it belongs.

remote queue. A queue belonging to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with *local queue*.

remote queue manager. To a program, a queue manager that is not the one to which the program is connected.

remote queue object. See *local definition of a remote queue*.

remote queuing. In message queuing, the provision of services to enable applications to put messages on queues belonging to other queue managers.

reply message. A type of message used for replies to request messages. Contrast with *request message* and *report message*.

reply-to queue. The name of a queue to which the program that issued an MQPUT call wants a reply message or report message sent.

report message. A type of message that gives information about another message. A report message can indicate that a message has been delivered, has arrived at its destination, has expired, or could not be processed for some reason. Contrast with *reply message* and *request message*.

requester channel. In message queuing, a channel that may be started remotely by a sender channel. The requester channel accepts messages from the sender channel over a communication link and puts the messages on the local queue designated in the message. See also *server channel*.

request message. A type of message used to request a reply from another program. Contrast with *reply message* and *report message*.

RESLEVEL. In MQSeries for OS/390, an option that controls the number of CICS user IDs checked for API-resource security in MQSeries for OS/390.

resolution path. The set of queues that are opened when an application specifies an alias or a remote queue on input to an MQOPEN call.

resource. Any facility of the computing system or operating system required by a job or task. In MQSeries for OS/390, examples of resources are buffer pools, page sets, log data sets, queues, and messages.

resource manager. An application, program, or transaction that manages and controls access to shared resources such as memory buffers and data sets. MQSeries, CICS, and IMS are resource managers.

Resource Recovery Services (RRS). An OS/390 facility that provides 2-phase syncpoint support across participating resource managers.

responder. In distributed queuing, a program that replies to network connection requests from another system.

resynch. In MQSeries, an option to direct a channel to start up and resolve any in-doubt status messages, but without restarting message transfer.

return codes. The collective name for completion codes and reason codes.

rollback. Synonym for *back out*.

RRS. Resource Recovery Services.

RTM. Recovery termination manager.

S

SAF. System Authorization Facility.

SDWA. System diagnostic work area.

security enabling interface (SEI). The MQSeries interface to which customer- or vendor-written programs that check authorization, supply a user identifier, or perform authentication must conform. A part of the MQSeries Framework.

SEI. Security enabling interface.

sender channel. In message queuing, a channel that initiates transfers, removes messages from a transmission queue, and moves them over a communication link to a receiver or requester channel.

sequential delivery. In MQSeries, a method of transmitting messages with a sequence number so that the receiving channel can reestablish the message sequence when storing the messages. This is required where messages must be delivered only once, and in the correct order.

sequential number wrap value. In MQSeries, a method of ensuring that both ends of a communication link reset their current message sequence numbers at the same time. Transmitting messages with a sequence number ensures that the receiving channel can reestablish the message sequence when storing the messages.

server. (1) In MQSeries, a queue manager that provides queue services to client applications running on a remote workstation. (2) The program that responds to requests for information in the particular two-program, information-flow model of client/server. See also *client*.

server channel. In message queuing, a channel that responds to a requester channel, removes messages from a transmission queue, and moves them over a communication link to the requester channel.

server connection channel type. The type of MQI channel definition associated with the server that runs a queue manager. See also *client connection channel type*.

service interval. A time interval, against which the elapsed time between a put or a get and a subsequent get is compared by the queue manager in deciding whether the conditions for a service interval event have been met. The service interval for a queue is specified by a queue attribute.

service interval event. An event related to the service interval.

session ID. In MQSeries for OS/390, the CICS-unique identifier that defines the communication link to be used by a message channel agent when moving messages from a transmission queue to a link.

shutdown. See *immediate shutdown*, *preemptive shutdown*, and *quiesced shutdown*.

signaling. In MQSeries for OS/390 and MQSeries for Windows 2.1, a feature that allows the operating system to notify a program when an expected message arrives on a queue.

single logging. A method of recording MQSeries for OS/390 activity where each change is recorded on one data set only. Contrast with *dual logging*.

single-phase backout. A method in which an action in progress must not be allowed to finish, and all changes that are part of that action must be undone.

single-phase commit. A method in which a program can commit updates to a queue without coordinating those updates with updates the program has made to resources controlled by another resource manager. Contrast with *two-phase commit*.

SIT. System initialization table.

storage class. In MQSeries for OS/390, a storage class defines the page set that is to hold the messages for a particular queue. The storage class is specified when the queue is defined.

store and forward. The temporary storing of packets, messages, or frames in a data network before they are retransmitted toward their destination.

subsystem. In OS/390, a group of modules that provides function that is dependent on OS/390. For example, MQSeries for OS/390 is an OS/390 subsystem.

supervisor call (SVC). An OS/390 instruction that interrupts a running program and passes control to the supervisor so that it can perform the specific service indicated by the instruction.

SVC. Supervisor call.

switch profile. In MQSeries for OS/390, a RACF profile used when MQSeries starts up or when a refresh security command is issued. Each switch profile that MQSeries detects turns off checking for the specified resource.

symptom string. Diagnostic information displayed in a structured format designed for searching the IBM software support database.

synchronous messaging. A method of communication between programs in which programs place messages on message queues. With synchronous messaging, the sending program waits for a reply to its message before resuming its own processing. Contrast with *asynchronous messaging*.

syncpoint. An intermediate or end point during processing of a transaction at which the transaction's protected resources are consistent. At a syncpoint, changes to the resources can safely be committed, or they can be backed out to the previous syncpoint.

sysplex. A multiple OS/390-system environment that allows multiple-console support (MCS) consoles to receive console messages and send operator commands across systems.

System Authorization Facility (SAF). An OS/390 facility through which MQSeries for OS/390 communicates with an external security manager such as RACF.

system.command.input queue. A local queue on which application programs can put MQSeries commands. The commands are retrieved from the queue by the command server, which validates them and passes them to the command processor to be run.

system control commands. Commands used to manipulate platform-specific entities such as buffer pools, storage classes, and page sets.

system diagnostic work area (SDWA). Data recorded in a SYS1.LOGREC entry, which describes a program or hardware error.

system initialization table (SIT). A table containing parameters used by CICS on start up.

SYS1.LOGREC. A service aid containing information about program and hardware errors.

T

task control block (TCB). An OS/390 control block used to communicate information about tasks within an address space that are connected to an OS/390 subsystem such as MQSeries for OS/390 or CICS.

task switching. The overlapping of I/O operations and processing between several tasks. In MQSeries for OS/390, the task switcher optimizes performance by allowing some MQI calls to be executed under subtasks rather than under the main CICS TCB.

TCB. Task control block.

temporary dynamic queue. A dynamic queue that is deleted when it is closed. Temporary dynamic queues are not recovered if the queue manager fails, so they can contain nonpersistent messages only. Contrast with *permanent dynamic queue*.

termination notification. A pending event that is activated when a CICS subsystem successfully connects to MQSeries for OS/390.

thread. In MQSeries, the lowest level of parallel execution available on an operating system platform.

time-independent messaging. See *asynchronous messaging*.

TMI. Trigger monitor interface.

trace. In MQSeries, a facility for recording MQSeries activity. The destinations for trace entries can include GTF and the system management facility (SMF). See also *global trace* and *performance trace*.

tranid. See *transaction identifier*.

transaction identifier. In CICS, a name that is specified when the transaction is defined, and that is used to invoke the transaction.

transmission program. See *message channel agent*.

transmission queue. A local queue on which prepared messages destined for a remote queue manager are temporarily stored.

trigger event. An event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

triggering. In MQSeries, a facility allowing a queue manager to start an application automatically when predetermined conditions on a queue are satisfied.

trigger message. A message containing information about the program that a trigger monitor is to start.

trigger monitor. A continuously-running application serving one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

trigger monitor interface (TMI). The MQSeries interface to which customer- or vendor-written trigger monitor programs must conform. A part of the MQSeries Framework.

two-phase commit. A protocol for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction. Contrast with *single-phase commit*.

U

undo/redo record. A log record used in recovery. The redo part of the record describes a change to be made to an MQSeries object. The undo part describes how to back out the change if the work is not committed.

unit of recovery. A recoverable sequence of operations within a single resource manager. Contrast with *unit of work*.

unit of work. A recoverable sequence of operations performed by an application between two points of consistency. A unit of work begins when a transaction starts or after a user-requested syncpoint. It ends either at a user-requested syncpoint or at the end of a transaction. Contrast with *unit of recovery*.

utility. In MQSeries, a supplied set of programs that provide the system operator or system administrator with facilities in addition to those provided by the MQSeries commands. Some utilities invoke more than one function.

Index

Sending your comments to IBM

Problem Determination Guide

GC34-5376-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form.
- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Readers’ Comments

Problem Determination Guide

GC34-5376-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States of America
Belgium	France	Israel	New Zealand	Spain	
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

1 Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories
Information Development Department (MP095)
Hursley Park,
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name ____________________
Company or Organization ____________________
Address ____________________
____
EMAIL ____________________
Telephone ____________________

1 Cut along this line

4 Fasten here with adhesive tape

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC34-5376-00